

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:

Zhao, Yali

Title:

Profit optimization of resource management for big data analytics-as-a-service platforms in cloud computing environments

Date:

2020

Persistent Link:

<https://hdl.handle.net/11343/243215>

Terms and Conditions:

Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.

Profit Optimization of Resource Management for Big Data Analytics-as-a-Service Platforms in Cloud Computing Environments

Yali Zhao

ORCID: 0000-0002-0697-5402

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

THE UNIVERSITY OF MELBOURNE

June 2020

Produced on archival quality paper.

Copyright © 2020 Yali Zhao

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Abstract

Discovering optimal resource management solutions to support data analytics to extract value from big data is an increasingly important research area. It is fair to say that the success of many organizations, companies, and individuals now relies heavily on data analytics solutions. Cloud computing greatly supports big data analytics by providing scalable resources based on user demand and supporting elastic resource provisioning in a pay-as-you-go model. Big data Analytics as a Service (AaaS) platforms provision AaaS to various domains as consumable services in an easy to use manner across cloud computing environments. AaaS platforms aim to deliver efficient data analytics solutions to benefit decision-making and problem solving in a wide range of application domains such as engineering, science, and government.

However, big data analytics solutions face a range of challenges: the dynamic nature of query requests; the heterogeneity of cloud resources; the different Quality of Service (QoS) requirements; the potential for lengthy data processing times and associated expensive resource costs and dealing with big data processing demands under potentially limited/constrained budgets, deadlines and/or data accuracies. The above challenges need to be tackled by efficient resource management solutions to support AaaS platforms to deliver reliable, cost-effective and fast AaaS. Optimal resource management solutions are essential for AaaS platforms to maximize profits and minimize query times while guaranteeing Service Level Agreements (SLAs) during AaaS delivery.

To tackle the above challenges, this thesis systematically studies profit optimization solutions to support AaaS platforms. Key contributions are made through a range of resource management solutions. These include admission control and resource scheduling algorithms that enable various problem scenarios where data needs to be processed under heterogeneous, constrained or limited budgets, deadlines, or accuracies with support of data splitting and/or data sampling-based methods to reduce data processing times and costs with potential accuracy trade-offs. These algorithms allow AaaS platforms to optimize profits and minimize query times through optimal resource management solutions, and thereby increase market share by maximizing query admissions and improve reputation by delivering SLA-supported AaaS solutions.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Yali Zhao, June 2020

Preface

This thesis has been composed using research works completed and published throughout the author's PhD program. The main contributions of the thesis are presented in Chapters 3-7. These are based on the following publications, where it is noted that in all of these papers, I have contributed more than 80% to the work they contain including proposing research problems, problem modelling and formulation, literature review, algorithm design and implementation, and performing all experiment analysis and paper writing.

Chapter 3 is based on the following publication:

Y. Zhao, R. N. Calheiros, G. Gange, K. Ramamohanarao, and R. Buyya, "SLA-based resource scheduling for big data analytics as a service in cloud computing environments," in *Proceedings of IEEE International Conference on Parallel Processing (ICPP)*, no. 1, pp. 510–519, 2015.

Chapter 4 is based on the following publication:

Y. Zhao, R. N. Calheiros, G. Gange, J. Bailey, and R. O. Sinnott, "SLA-based profit optimization for resource scheduling of big data analytics-as-a-service in cloud computing environments," *IEEE Transactions on Cloud Computing (TCC)*, pp. 1-18, 2018.

This paper is extended from the following publication:

Y. Zhao, R. N. Calheiros, J. Bailey, and R. O. Sinnott, "SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments," in *Proceedings of IEEE International Conference on Big Data (Big Data)*, pp. 432-441, 2016.

Chapter 5 is based on the following publication:

Y. Zhao, R. N. Calheiros, A.V. Vasilakos, J. Bailey, and R. O. Sinnott, “Profit maximization and time minimization admission control and resource scheduling for cloud-based big data analytics-as-a-service platforms,” in *Proceedings of International Conference of Web Services (ICWS)*, pp. 26-47, 2019.

Chapter 6 is based on the following publication:

Y. Zhao, R. N. Calheiros, A. V. Vasilakos, J. Bailey, and R. O. Sinnott, “SLA-aware and deadline constrained profit optimization for cloud resource management in big data analytics-as-a-service platforms,” in *Proceedings of IEEE International Conference on Cloud Computing (CLOUD)*, pp. 146-155, 2019.

Chapter 7 is based on the following publication:

Y. Zhao, A.V. Vasilakos, J. Bailey, and R. O. Sinnott, “Profit optimization for splitting and sampling based resource management in big data analytics-as-a-service platforms in cloud computing environments,” in *Proceedings of International Conference of e-Science (e-Science)*, pp. 1-12, 2019.

Acknowledgements

First, I would like to thank my principle supervisor Prof. Sinnott for encouragement and support throughout the journey. He helped me realize the importance of formulating the right research problems and attention to details.

I would like to thank my supervisor Prof. Bailey who guided and supported me with his depth of domain knowledge, valued insights and infectious enthusiasm.

I would like to express my deepest appreciation for my supervisor Dr. Calheiros who has always been supportive and who offered tremendous help to me throughout this journey and directed my path with his unconditional support.

I would like to thank my supervisor Prof. Vasilakos who motivated me to work on research challenges and continuously to improve the research works with meaningful impacts.

I would like to thank my committee chair Prof. Vetere for encouraging me towards my PhD goals and being supportive with insightful comments.

I am grateful for all colleagues and anonymous reviewers who reviewed my papers and provided meaningful feedback to improve the works contained therein.

I am appreciative of my friends who supported me and cared for me during the journey.

I would like to thank my family who have been always supportive and caring and helped me conquer challenges and difficulties. I am deeply appreciative of their unconditional love and encouragement throughout the journey to pursue my dream.

A PhD is a meaningful journey and a starting point for opening a new chapter to make changes that matter and make contributions that have meaningful impact to benefit both scientific research and society.

Thank you all!

Yali Zhao

June 2020

To My Family

Content

LIST OF FIGURES	XVII
------------------------	-------------

LIST OF TABLES	XIX
-----------------------	------------

1	INTRODUCTION	1
1.1	RESEARCH MOTIVATION AND SCOPE	1
1.2	RESEARCH PROBLEM	7
1.3	THESIS CONTRIBUTION	8
1.4	THESIS ORGANIZATION	11
2	BACKGROUND AND RELATED WORK	15
2.1	INTRODUCTION	15
2.1.1	<i>Big Data</i>	15
2.1.2	<i>Cloud Computing</i>	16
2.1.3	<i>Cloud-based Big Data AaaS Platforms</i>	19
2.1.4	<i>Resource Management for Cloud-Based Big Data AaaS Platforms</i>	23
2.2	RELATED WORK	24
2.2.1	<i>Cost and Time Optimization of Resource Management</i>	24
2.2.2	<i>Cost Optimization of Resource Management</i>	25
2.2.3	<i>Time Optimization of Resource Management</i>	28
2.2.4	<i>Resource Management with Other Objectives</i>	28
2.3	CONTEXTUALIZING THE THESIS CONTRIBUTION	31
3	SLA-BASED PROFIT ENHANCEMENT FOR RESOURCE MANAGEMENT	33
3.1	INTRODUCTION	35
3.2	ARCHITECTURE AND MODELS	36
3.2.1	<i>Architecture</i>	36
3.2.2	<i>Models</i>	38
3.3	ADMISSION CONTROL AND RESOURCE SCHEDULING	39
3.3.1	<i>Admission Control Algorithm</i>	39
3.3.2	<i>Scheduling Algorithms</i>	39
3.4	PERFORMANCE EVALUATION	47
3.4.1	<i>Resource Configuration</i>	48
3.4.2	<i>Workload</i>	48
3.4.3	<i>Result Analysis</i>	49
3.5	RELATED WORK	57
3.6	CONCLUSIONS AND FUTURE WORK	59
4	SLA-BASED PROFIT OPTIMIZATION OF RESOURCE MANAGEMENT	61
4.1	INTRODUCTION	63
4.2	PROBLEM STATEMENT	68
4.2.1	<i>Overview of the AaaS Environment</i>	68
4.2.2	<i>Architecture of the AaaS Platform</i>	68
4.2.3	<i>Problem Modelling</i>	70
4.2.4	<i>Problem Formulation</i>	71
4.3	ADMISSION CONTROL AND RESOURCE SCHEDULING	76

4.3.1	<i>Admission Control</i>	77
4.3.2	<i>Resource Scheduling</i>	79
4.4	PERFORMANCE EVALUATION	85
4.4.1	<i>Experiment Setup</i>	85
4.4.2	<i>SLA Based Resource Scheduling</i>	87
4.4.3	<i>SLA and Data Splitting Based Resource Scheduling</i>	93
4.5	RELATED WORK	100
4.6	CONCLUSION AND FUTURE DIRECTIONS	103
5	SLA-AWARE AND DATA SAMPLING BASED PROFIT OPTIMIZATION OF RESOURCE MANAGEMENT	105
5.1	INTRODUCTION	107
5.2	PROBLEM STATEMENT	109
5.3	ADMISSION CONTROL AND RESOURCE SCHEDULING	115
5.3.1	<i>Admission Control</i>	115
5.3.2	<i>Resource Scheduling</i>	117
5.4	PERFORMANCE EVALUATION	122
5.5	RELATED WORK	132
5.6	CONCLUSION AND FUTURE WORKS	134
6	SLA-AWARE AND DATA SPLITTING BASED PROFIT OPTIMIZATION OF RESOURCE MANAGEMENT	135
6.1	INTRODUCTION	137
6.2	PROBLEM STATEMENT	139
6.2.1	<i>Problem Model</i>	139
6.2.2	<i>Problem Statement</i>	140
6.3	ADMISSION CONTROL AND RESOURCE SCHEDULING	147
6.3.1	<i>Admission Control</i>	147
6.3.2	<i>Resource Scheduling</i>	149
6.4	PERFORMANCE EVALUATION	152
6.4.1	<i>Experiment Setup</i>	152
6.4.2	<i>SLA-Aware and Data Splitting-Based Scheduling</i>	155
6.5	RELATED WORK	162
6.6	CONCLUSION AND FUTURE WORKS	164
7	SLA-AWARE DATA SPLITTING AND SAMPLING BASED PROFIT OPTIMIZATION OF RESOURCE MANAGEMENT	165
7.1	INTRODUCTION	167
7.2	PROBLEM STATEMENT	170
7.3	RESOURCE SCHEDULING	178
7.3.1	<i>Splitting and Sampling (SS) Based Profit Optimization Algorithm</i>	178
7.3.2	<i>Profit Optimization (PO) Algorithm</i>	181
7.3.3	<i>SPLITting (SP) Based Profit Optimization Algorithm</i>	182
7.3.4	<i>Sampling (SA) Based Profit Optimization Algorithm</i>	182
7.4	PERFORMANCE EVALUATION	182
7.4.1	<i>Experiment Setup</i>	182
7.4.2	<i>Admission Control and SLA Guarantee</i>	183
7.4.3	<i>Profit Maximization</i>	184
7.4.4	<i>Cost Saving</i>	186
7.4.5	<i>Resource Configuration</i>	187
7.4.6	<i>Algorithm Performance for BDAAs</i>	189
7.4.7	<i>Algorithm Running Time (ART) Analysis</i>	192
7.5	RELATED WORK	195
7.6	CONCLUSION AND FUTURE WORKS	196

8	CONCLUSIONS AND FUTURE DIRECTIONS.....	199
8.1	CONCLUSIONS	199
8.2	FUTURE DIRECTIONS.....	202

List of Figures

Figure 3.1: Architecture of the AaaS Platform.....	37
Figure 3.2: Resource Cost of AGS, AILP, and ILP.	51
Figure 3.3: Profit of AILP and AGS.	53
Figure 3.4: Profit and Resource Cost of AILP and AGS.	54
Figure 3.5: Profit and Resource Cost of BDAAAs for SI=20.....	55
Figure 3.6: C/P of AILP and AGS.....	56
Figure 3.7: ART of AILP and AGS.....	56
Figure 4.1: Overview of the AaaS Environment.	66
Figure 4.2: Architecture of the AaaS Platform.....	67
Figure 4.3: Median Profits and Resource Costs of PO, HGS, AILP, and AGS Algorithms...88	
Figure 4.4: Profits and Resource Costs of PO, HGS, AGS, and AILP Algorithms.	88
Figure 4.5: Profits and Resource Costs of Different BDAAAs.	89
Figure 4.6: Profits and Resource Costs of PO and HGS Algorithms for Single Schedules....	91
Figure 4.7: ART of PO, HGS, AILP, and AGS Algorithms.	92
Figure 4.8: Admission Rates of PO and SPO Algorithms.....	93
Figure 4.9: Resource Costs and Profits of PO and SPO Algorithms.....	94
Figure 4.10: Profits and Resource Costs of PO and SPO Algorithms for Single Schedules. .95	
Figure 4.11: Profits and Resource Costs of PO and SPO Algorithms for BDAAAs with SI=80.	96
Figure 4.12: Workload Running Times of PO and SPO Algorithms.	97
Figure 4.13: ART of PO and SPO Algorithms.....	97
Figure 5.1: Admission Rates of SAPO and PO.	123
Figure 5.2: Profits of SAPO and PO.	124
Figure 5.3: Resource Costs of SAPO and PO.	125
Figure 5.4: Resource Configurations of SAPO and PO.	126
Figure 5.5: Query Number of Data Sampling and Non Sampling Processing of SAPO and PO.	126
Figure 5.6: Query Number of Sampling and Non Sampling Processing for Each BDAA. ..	127
Figure 5.7: Approximate Query Processing Number for Each BDAA.	128
Figure 5.8: Query Acceptance Number for BDAAAs.	129
Figure 5.9: Profits of SAPO and PO for Different BDAAAs.	129
Figure 5.10: Resource Costs of BDAAAs of SAPO and PO for Realtime and SI=1.	130
Figure 5.11: Accuracy Analysis.	130
Figure 5.12: ART Analysis.....	131
Figure 6.1: Dataflow Example for Splitting-Based Query Processing.....	144
Figure 6.2: Admission Rates of SPPO and PO.....	153
Figure 6.3: Profits of SPPO and PO.	154
Figure 6.4: Query Processing Methods of SPPO and PO.	155
Figure 6.5: Query Profits of SPPO and PO.	156

Figure 6.6: VM Configuration of SPPO and PO.	156
Figure 6.7: Resource Costs of SPPO and PO.	157
Figure 6.8: Query Resource Costs of SPPO and PO.	158
Figure 6.9: Query Number for Splitting and Non Splitting Processing of SPPO and PO for BDAAs.	158
Figure 6.10: Resource Configurations of SPPO and PO for BDAAs.	159
Figure 6.11: BDAA Resource Costs of SSPO and PO.....	160
Figure 6.12: BDAA Profits of SPPO and PO.....	160
Figure 6.13: ART of SPPO and PO.....	161
Figure 7.1: Query Admission Rates.	184
Figure 7.2: Profits.	185
Figure 7.3: Data Processing Methods of Queries.	186
Figure 7.4: Query Profits.	187
Figure 7.5: Resource Costs.....	188
Figure 7.6: Resource Configurations.....	189
Figure 7.7: Query Resource Costs.....	190
Figure 7.8: Admitted Query Number for BDAAs.....	191
Figure 7.9: Resource Configurations of BDAAs.	191
Figure 7.10: Resource Costs of BDAAs.....	193
Figure 7.11: Profits of BDAAs.....	193
Figure 7.12: Algorithm Running Time.....	194

List of Tables

Table 3.1: Definition of Symbols	41
Table 3.2: VM Configuration	48
Table 3.3: Query Number Information.....	50
Table 3.4: Resource Configuration.....	52
Table 3.5: Related Work.....	58
Table 4.1: Definition of Symbols	74
Table 4.2: Query Admission Rates and SLA Guarantees	86
Table 4.3: Resource Configuration.....	90
Table 4.4: Query Admission and SLA Guarantee	94
Table 4.5: Resource Configuration.....	96
Table 4.6: Related Works	102
Table 6.1: Definition of Symbols	141
Table 7.1: Symbol Definitions	172

1 Introduction

1.1 Research Motivation and Scope

As we step into the big data era [1], a tremendous amount of data is now being produced at an ever-accelerating rate from a myriad of sources such as IoT [159] sensors, smart mobile devices, social media platforms amongst many other sources. These cover all aspects of society including health, commerce, government and education. Big data typically comprises massive volumes of structured, unstructured, and semi-structured data that exceeds the management and processing capacity of traditional resources and/or processing techniques [4, 5].

Big data [2] is an emerging trend that many organizations and companies are currently facing. The need for advanced technologies to support larger-scale data production, processing, management, and storage is clear. However, big data can only create value through the help of data analytics technologies to derive insights from such big datasets. It is fair to say that the success of many organizations, companies, and individuals now relies heavily on big data analytics solutions.

Key enablers of big data are the increased capability of storage and computing resources enabled by recent advances in cloud computing and the increased availability of data. The ease of access to data supported by various companies such as Amazon, Google, IBM, and Facebook have made big data solutions a reality [6].

Big data analytics is usually associated with cloud computing technologies [7]. Cloud computing [8] directly supports and benefits big data analytics by providing a computing paradigm to dynamically provision resources for big data analytics solutions required to cope with a varying number of requests from users. Cloud computing can be delivered in many ways: (1) Infrastructure-as-a-Service (IaaS), such as Amazon EC2, which offer on-demand virtualized resources, i.e. Virtual Machines (VMs) [156] and containers; (2) Platform-as-a-Service (PaaS), such as Google App Engine, which manage many of the underlying infrastructure resources allowing users to deploy and run applications, through to (3) Application/Software-as-a-Service (SaaS), such as IBM Social Media Analytics [9] that are provided to users through web browser portals [6].

As the volume of big data increases to levels that exceed the storage [160] and processing capabilities of individual computers, clouds allow to automatically and

elastically provision on-demand resources to process datasets for timely decision making as required by users and businesses. Cloud computing offers scalable computing resources with Service Level Agreement (SLA) guarantees. An SLA [10] is an agreement that defines the metrics, expected QoS, and penalties that may occur during service delivery. These are established between service users and providers. It is critical to guarantee SLAs to provide high quality of services [161]. Cloud resources are typically provisioned in a pay-per-use model where users only pay for the resources used. Hitherto, big data analytics for small enterprises and individuals has not been easy or possible, but with the growth of cloud computing and tooling this is now changing.

Analyzing big data to extract potential insights from the data is essential for individuals, organizations, and governments to make better decisions, i.e. product trend prediction, business strategy making, and disaster prediction and management. The value that can be extracted from big data has motivated users from various application domains to explore data analytics techniques and solutions to gain benefits in decision making and problem solving [11]. However, providing big data analytics solutions to extract insights from big data still faces numerous challenges.

Big data analytic solutions are often expensive due to the demand for large data storage systems to store voluminous amounts of data. Moreover, big data analytics requires large-scale and often high-performance computing resources to process ever-increasing data volumes that can result in considerable expense and processing times.

Big Data Analytics Applications (BDAAAs) are often targeted and customized to specific application domains [12]. Licenses for BDAAAs can be expensive. Customized BDAAAs are often used by large enterprises. However, it can be difficult to provision resources for BDAAAs with fluctuating resource requirements whilst aiming to minimize the resource costs. As a result, systematic adoption of data analytics solutions is limited to larger enterprises or major research projects, and they are not suitable for occasional or individual usage [11].

The value that can be extracted from big data motivates users to explore data analytics solutions for improved problem solving [3]. Optimal resource management solutions to extract values from big data to benefit decision making across a wide range of application domains such as engineering, science, and healthcare is important for research and society alike. This thesis focuses on delivering SLA-aware profit

optimization resource management solutions for AaaS platforms to deliver cost-effective, timely, and reliable AaaS to tackle big data analytics challenges.

Managing resources for BDAAAs requires expert knowledge. For example, BDAAAs often need to be deployed on various machines for parallel processing on large datasets in order to reduce the time for generating data analytics solutions while resources should be scaled up and down for varied resource demands in order to save energy and costs. The lack of such expertise makes resource management for BDAAAs difficult for many users.

The above challenges drive the need for an Analytics as a Service (AaaS) platform to provide optimal big data analytics solutions. To provision online big data AaaS to users in various domains, a general purpose AaaS platform is required to deliver on-demand and easy-to-use AaaS with SLA guarantees in a timely manner and with controllable costs [13].

Cloud-based big data Analytics-as-a-Service (AaaS) platforms need to serve on-demand big data analytics requests and provision AaaS solutions to various domains of users as automatic services that are easy to use and at low cost. Big data AaaS platforms need to deliver data analytics as consumable services in a pay-as-you-go model with SLA guarantees supported by the cloud computing environments.

AaaS platforms need to broker commodity resources, such as VMs and containers, from cloud infrastructure resource providers and BDAAAs from third-party application providers. BDAAAs that typically process read-only queries and do not handle data consistency and security are common applications [14]. AaaS platforms supporting automatic resource provisioning and on-demand AaaS delivery in a pay-per-use basis can significantly benefit individuals and small enterprises with occasional usage patterns. They should significantly reduce the AaaS costs to enable users to only pay for AaaS they use without the need to purchase expensive hardware and software. AaaS platforms should ideally hide the complexity of resource management for users with limited expertise in distributed resource managements.

AaaS are often delivered as on-demand services with resources automatically managed based on the demands of data analytics requests/tasks. In this thesis, the focus is on data analytics requests that query large datasets leaving out data security and

consistency issues. Quality of Service (QoS) [15] requirements of queries need to be satisfied by the delivered AaaS with SLA guarantees.

Resource management serves as the core of AaaS platforms to automate resource provisioning, query processing, and coordinating all components to deliver high quality AaaS with SLA guarantees. Optimal resource management solutions for AaaS platforms are essential to maximize profits for providers and minimize processing times by optimizing resource and query scheduling for users. This should balance increased market share by admitting more queries and improving user satisfaction by delivering more reliable AaaS with SLA guarantees [16].

Resource management for AaaS platforms is essential as big data analytics require large-scale computing, which can incur high resource costs. Resource management enables AaaS platforms to provision resources to execute queries with guaranteed SLAs based on QoS requirements such as budgets, deadlines and accuracies. This allows users to control analytic costs and obtain timely and reliable results for efficient decision-making. Resource management of AaaS platforms must therefore tackle the following challenges:

- Resource management must tackle the challenge of dynamic nature of query requests and the heterogeneity of cloud resources. Queries submitted by users are highly dynamic and can be stochastic in nature. Resource requirements of online queries can thus fluctuate over time, while cloud resources are heterogeneous and offer different configurations at varied costs. Automatic resource management is expected to provision heterogeneous resources to process potentially, highly dynamic query workloads.
- Resource scheduling requires high scalability in order to provision resources for query execution with SLA guarantees while providing cost-efficient solutions to maximize profits (for providers). Over-provisioning of cloud resources in AaaS platforms incurs high resource costs that can substantially decrease profits. However, under provisioning of cloud resources to BDAAAs can cause SLA violations. Therefore, auto-scaling of cloud resources is a necessity for resource management.
- The different QoS requirements of queries requires query admission to determine the acceptance and rejection of queries. Effective admission control

should only admit queries satisfying QoS requirements to meet given SLAs and avoid user dissatisfaction caused by SLA violations.

- Big data analytics requires large-scale computing and storage resources that can incur considerable resource costs. It is a necessary to provide cost-effective resource scheduling solutions to enable users to obtain query results under controllable costs, especially for users with limited budgets.
- Big data analytics for data with large volumes can take considerable time. It is essential to provide efficient resource management and task scheduling solutions that are capable of establishing large-scale computing environments and return query results before user-specified deadlines for fast and efficient decision-making. This is especially important for application domains needing timely decisions, such as disaster management and sales forecasting. Moreover, query response times are a major performance indicator of service delivery and these should be minimized for better user experience. It is a necessity to provide time-effective resource scheduling solutions to enable users to obtain timely query results under controllable deadlines, especially for application domains where rapid decision making is needed.
- The accuracy of queries may need to trade off with performance in order to deliver meaningful data analytics results in a timely manner. Different application domains may demand the highest possible accuracy while others can trade-off query accuracy to reduce resource costs and query response times. For queries requiring full accuracy, the complete datasets need to be processed under budgets and deadline constraints for reliable decision making. For queries whose accuracy can be traded-off, data processing techniques, e.g. sampling, can to be applied to maximize profits and admit queries where the absolute accuracy of results is not as important.
- Obtaining optimal resource management solutions to maximize profits while minimizing query response times and guaranteeing SLAs under various constraints represents a nontrivial multi-objective optimization problem since it must tackle dynamic query requests, heterogeneous cloud resource and BDAs, complex optimization parameters, and various query and resource constraints.

To seek optimal resource scheduling solutions, the scheduling problem requires accurate modelling and formulation.

- To tackle the challenges of big data processing under constraints of tight/limited budgets, deadlines or accuracies, resource management should enable efficient adoption of data processing techniques to deliver reliable AaaS delivery with faster response times and minimised resource costs. Data splitting and sampling-based resource management solutions can help optimize profits while satisfying budgets, deadlines, and/or accuracy constraints of queries with support for parallel and/or approximate query processing on very large datasets. Not being able to admit data analytics requests under tight budgets and deadlines can cause a reduction of profit, decrease user satisfaction, and lead to loss of customers. This can be avoided with efficient resource management solutions.

Motivation and Research Focus:

There is no solution in the literature that is able to tackle the above research challenges in offering comprehensive SLA-aware profit optimization solutions with automated resource management and deliver AaaS with accuracy, budget, and deadline requirements. The motivation of this thesis is to provide optimal and comprehensive resource management solutions to maximize profits and minimize query times while delivering timely, reliable, and cost-effective AaaS to users. We aim to serve the interests of AaaS providers to enhance profits by proposing optimal resource management solution and increase market share by maximizing the number of query admissions, and improve their reputation by delivering satisfactory services to users with associated SLA guarantees. To support the AaaS platform, our research focuses on proposing efficient and automatic admission control and resource scheduling algorithms to tackle big data challenges that can maximize profits for AaaS platforms while minimizing query processing times and deliver AaaS satisfying user requirements. We provide SLA guarantees on budgets, deadlines, and/or accuracy requirements during AaaS delivery to support timely, cost-effective, and reliable decision making and problem solving.

1.2 Research Problem

This thesis focuses on resource management solutions to optimize profits and deliver AaaS with minimized query times and SLA guarantees across cloud computing environments. Our research works focus on the following five problem scenarios.

Problem Scenario 1: *How to provide near-optimal resource management solutions to enhance profits that*

- *maximize utilization of existing cloud resources to execute queries;*
- *scale resources up when resources are under-provisioned with minimized costs;*
- *scale resources down when resources are over-provisioned to reduce costs, and*
- *execute queries with SLA guarantees on budget and deadline requirements?*

Problem Scenario 2: *How to provide optimal resource management solutions to maximize profits that*

- *select optimal resource configurations to execute queries;*
- *provide heuristic solutions to offer near-optimal solutions to reduce runtime of optimization algorithms;*
- *provide backup heuristic solutions to deliver results in a time-efficient manner where necessary;*
- *process representative data to reduce data processing times and costs, and*
- *guarantee SLAs on budgets and deadlines?*

Problem Scenario 3: *How to provide reliable resource management solutions to maximize profits that*

- *automatically and elastically provision cloud resources to execute queries with minimized data processing times and costs;*
- *support big data processing where there are insufficient deadlines or budgets through data sampling-based methods satisfying accuracy constraints to support reliable decisions, and*
- *provide SLA guarantees on different budgets, deadlines, and accuracy constraints associated with queries?*

Problem Scenario 4: *How to provide time-efficient resource management algorithms to optimize profits that*

- *deliver timely responses for application domains require rapid decision-making where fully processing of big data is necessary for accurate results;*
- *support queries under limited deadlines with sufficient budgets where data splitting-based methods can be applied to process split datasets in parallel to speed up AaaS delivery;*
- *automatically and efficiently schedule resources to minimize costs and queries times, and*
- *deliver timely and cost-efficient AaaS solutions with SLA guarantees on budget and deadline requirements?*

Problem Scenario 5: *How to provide cost-effective, timely, and reliable resource management algorithms to optimize profits that*

- *support different big data processing scenarios where data analytic requests have heterogeneous budget, deadline, and accuracy requirements;*
- *support queries under insufficient/limited deadlines where parallel processing can be applied to speed up data processing;*
- *support queries under insufficient budgets or deadlines where data sampling can be applied to reduce data processing times and costs;*
- *prioritize fully accurate processing of big data where budgets and deadlines can be satisfied whilst enabling query accuracy trade-offs when necessary to tackle cost and time challenges, and*
- *deliver AaaS with SLA guarantees on budget, deadline, and accuracy requirements?*

1.3 Thesis Contribution

This thesis explores profit optimization resource management solutions for AaaS platforms to deliver cost-effective, timely, and reliable AaaS to tackle big data analytics challenges. We propose admission control and resource scheduling algorithms to enable big data processing under deadline, budget, and/or accuracy constraints

associated with data analytics requests. Our proposed algorithms offer optimized resource management solutions that enable SLA-guarantees on query execution and resource provisioning to maximize the profits of AaaS platforms and minimize the query times for users. To provide comprehensive and optimal resource management solutions, we focus on novel admission control and resource scheduling algorithms to support various big data processing scenarios that represent the contributions of this thesis.

Chapter 3 offers a solution for **Problem Scenario 1**.

The major contributions of this chapter are automatic and scalable admission control and resource scheduling algorithms that admit queries based on QoS requirements and delivers analytic solutions to users with SLA guarantees on budgets and deadlines. The proposed resource management solutions create higher profits by providing cost-effective resource scheduling solutions to enlarge market share by increasing query admission rates and improve reputation by delivering services to users with SLA guarantees. This chapter tackles multi-objective optimization problems to maximize profits and minimize query times with accurate modeling and efficient solutions that deploy BDAAAs on large-scale resource for big data processing. These are used to obtain timely analytic solutions while automatically scaling resources up and down depending on the resource demands with the aim to reduce resource consumption and hence cost.

Chapter 4 offers solution for **Problem Scenario 2**.

The major contributions of this chapter are automatic and scalable resource scheduling algorithms to maximize profits for AaaS platforms while minimizing query processing times and delivering AaaS services to users with SLA guarantees on budgets and deadlines to ultimately allow timely responses and controllable costs. The proposed algorithms tackle big data processing challenges under limited budgets and times by applying data splitting-based scheduling methods that enable users to trade-off accuracy for faster response times and less resource costs. The proposed profit optimization resource scheduling algorithms can efficiently and automatically schedule heterogeneous cloud resources to deal with highly dynamic query workloads; support data-aware scheduling; tackle time and cost challenges for big data processing, and

allow users to obtain reliable data analytics solutions in a timely manner with controllable costs and SLA guarantees.

Chapter 5 provides solution for **Problem Scenario 3**.

The major contributions of this chapter are efficient and effective admission control and resource scheduling algorithms to elastically and automatically provision cloud resources to schedule queries that serve the objectives of profit maximization for AaaS providers and query time minimization for users while guaranteeing SLAs. To support queries on very large datasets that cannot be processed under limited budgets or deadlines where accuracy can be traded-off, the proposed algorithms apply data sampling-based admission and scheduling methods to deliver timely AaaS solutions with accuracy, deadline, and budget guarantees for reliable decision making under controllable budgets and deadlines.

Chapter 6 provides solution for **Problem Scenario 4**.

The major contributions of this chapter are efficient and automatic admission control and resource scheduling algorithms to elastically provision cloud resources to schedule queries that serve the objectives of profit maximization for AaaS providers and query time minimization for users while guaranteeing SLAs. To support queries on very large datasets for applications where fast decision making is required under limited deadlines where full accuracies are required for reliable decision making, the proposed algorithms apply data splitting-based admission and scheduling methods that deliver timely AaaS solutions to speed up big data analytics through parallel processing thereby offering timely and reliable decision making under controllable budgets and deadlines.

Chapter 7 provides solution for **Problem Scenario 5**.

The major contributions of this chapter are automatic and scalable admission control and resource scheduling algorithms that maximize the profits for AaaS platforms and minimize query response times with efficient query execution solutions and optimal resource utilization. To enable big data processing constrained by deadlines and limited budgets, the proposed algorithms support splitting and sampling-based admission control and resource scheduling for parallel and/or approximate processing. These

significantly reduce data processing times and resource costs and offer reliable AaaS under accuracy constraints. The proposed algorithms admit queries whose QoS requirements can be satisfied to deliver services with SLAs guarantees on budgets, deadlines and accuracy constraints, whilst supporting flexible query dependencies and data locality-aware scheduling mechanisms. The proposed admission control and resource scheduling algorithms provide a comprehensive solution for AaaS platforms to tackle big data processing challenges including dynamic query requests, heterogeneity of cloud resources, different QoS requirements, various data processing times, expensive resource costs, and big data processing demands under limited budgets and deadlines.

1.4 Thesis Organization

The core chapters of the thesis are derived from numerous conference and journal papers published during my PhD study. The thesis is structured as follows:

- **Chapter 1** presents the research motivation and scope, research problems and objectives, research contribution, and the organization of the thesis;
- **Chapter 2** introduces the background and related work regarding profit optimization resource management for big data AaaS platforms in cloud computing environments;
- **Chapter 3** presents SLA-based profit enhancement algorithms that maximize resource utilization, minimize resource costs, and apply two stage scaling method that automatically manage resources based on dynamic query demands.

This chapter is derived from:

- **Y. Zhao**, R. N. Calheiros, G. Gange, K. Ramamohanarao, and R. Buyya, “SLA-based resource scheduling for big data analytics as a service in cloud computing environments,” in *Proceedings of IEEE International Conference on Parallel Processing (ICPP)*, pp. 510–519, 2015.
- **Chapter 4** provides SLA-based profit optimization algorithms that effectively admit queries, select optimal resource configurations, minimize query times, and process data representatives to reduce data processing times and costs. This chapter is derived from:

- **Y. Zhao**, R. N. Calheiros, G. Gange, J. Bailey, and R. O. Sinnott, “SLA-based profit optimization for resource scheduling of big data analytics-as-a-service in cloud computing environments,” *IEEE Transactions on Cloud Computing* (TCC), pp. 1-18, 2018 (in Press). This paper is an extension of: **Y. Zhao**, R. N. Calheiros, J. Bailey, and R. O. Sinnott, “SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments,” in *Proceedings of IEEE International Conference on Big Data* (Big Data), pp. 432-441, 2016.
- **Chapter 5** presents SLA-aware and sampling-based profit optimization algorithms that enable reliable big data analytics under constrained budgets, deadlines, and accuracies. This chapter is derived from:
 - **Y. Zhao**, R.N. Calheiros, A.V. Vasilakos, J. Bailey, and R. O. Sinnott, “Profit maximization and time minimization admission control and resource scheduling for cloud-based big data analytics-as-a-service platforms,” in *Proceedings of 2019 International Conference of Web Services* (ICWS), pp. 26-47, 2019.
- **Chapter 6** provides SLA-aware and data splitting-based resource management algorithms to support reliable and fast results where full query accuracy is required under hard deadlines. This chapter is derived from:
 - **Y. Zhao**, R. N. Calheiros, A. V. Vasilakos, J. Bailey, and R. O. Sinnott, “SLA-aware and deadline constrained profit optimization for cloud resource management in big data analytics-as-a-service platforms,” in *Proceedings of 2019 IEEE International Conference on Cloud Computing* (CLOUD), pp. 146-155, 2019.
- **Chapter 7** supports SLA-aware splitting and sampling-based profit optimization algorithms that support multiple big data processing scenarios and provide comprehensive solutions to support query processing under limited/insufficient budget, deadline, and accuracy constraints to enable reliable, fast, and cost-effective AaaS delivery. This chapter is derived from:
 - **Y. Zhao**, A.V. Vasilakos, J. Bailey, and R. O. Sinnott, “Profit optimization for splitting and sampling based resource management in

big data analytics-as-a-service platforms in cloud computing environments,” in *Proceedings of 2019 International Conference of e-Science* (e-Science), pp. 156-167, 2019.

- **Chapter 8** concludes the thesis and highlights potential future research directions and works.

2 Background and Related Work

2.1 Introduction

Big data analytics is essential to gain insight from big data to benefit decision making and problem solving for various application domains and industries. Cloud computing forms a cornerstone in deploying BDAAAs to elastically provision resources based on user requirements in a pay-as-you-go model. BDAAAs often cater for specific domains and are usually expensive. It is necessary to have a general AaaS platform that provisions data analytics services to users in multiple domains with SLA guarantees at controllable costs and in a timely and easy-to-use manner. Resource management is a core function of AaaS platforms and should maximize profits and minimize query times while guaranteeing SLAs. In order to support AaaS platforms in delivering fast, cost-effective and reliable AaaS, in this chapter we focus on the background and related research work focused on resource management for AaaS platforms.

It is noted that each chapter (Chapters 3-7) includes references, materials, and summaries related to the state of the art for the individual chapters, hence this chapter focuses predominantly on the background to cloud computing and more general related challenges with the deeper insights and related work included in the reference sections of each paper/chapter.

2.1.1 Big Data

Big data typically refers to massive volumes of structured, unstructured, semi-structured, or mixed types of data. These are typically difficult to manage and analyze using traditional database techniques [4, 17]. Big data is often characterized by the *V* characteristics: *volume* (massive amounts), *velocity* (fast speed of production and/or acquisition), *variety* (various modalities and heterogeneity), *veracity* (uncertainties around the source of truth), and *value* (the actual merit of the data) [11]. Data-intensive science [18] emerged as the fourth scientific paradigm extending empirical science, theoretical science and computational science. Data-intensive science and especially data-intensive computing aims to provide the tools to handle big data problems [19].

Big data analytics typically involves the process of examining big data to uncover hidden patterns, unknown correlations and other information that can be used to make

better decisions [20]. Data analytics often plays a significant role in decision making and problem solving for various data and application domains. It allows complex computations to be executed over the data to discover knowledge, gain insights, and generally extract value/knowledge from the data. Data analytics generally comprises a sequence of tasks such as data cleansing, data splitting, data sampling, data analysis, and data integration that ranges from simple data analytic queries to complex tasks such as machine learning algorithms and workflows [21]. Big Data Analytic Applications (BDAAAs) are designed to discover knowledge, extract insights, and create value from analyzing big data.

AaaS platforms deploy BDAAAs to analyse large amounts of data and ultimately identify the hidden patterns and unknown correlations in the data to derive value, knowledge, and/or insights that can be used to guide decision making. Data analytics methodologies are enabled by a new generation of technologies, such as in-memory processing, parallel processing [153], stream-based processing. Memory requirements and multi-tenancy are important factors for AaaS systems to handle data-intensive computing and deliver AaaS solutions to uses in various domains. Cloud computing and cloud architectures are designed with BDAAAs in mind.

Analyzing and gaining insight of big data can revolutionize many application domains [22]. There is a growing demand for AaaS solutions for better decision making and problem solving. The global AaaS market size was valued at \$9.62 billion in 2018 and is predicted to reach \$126.48 billion by 2026 based on an Allied Market Research review [73]. AaaS solutions allow users and organisations in various domains to gain competitive advantage and make better business decisions, e.g. strategic or operational decisions that can lead to improved efficiency, increased customer satisfaction and enhanced revenue.

2.1.2 Cloud Computing

Cloud computing environments are well-suited to process big data analytics requests as the processing can be partitioned and deployed on cloud. With the rapid progress of information and communication technologies over the past few decades, computing has become the fifth utility to provide services to satisfy society's needs, augmenting the

other four utilities of water, electricity, gas and telephony [23]. Cloud computing supports networked access to a shared resource pool where large-scale and often heterogeneous computing resources are provided [24, 25]. Users can get access to the computing resources through the Internet anytime, anywhere around the world in a convenient, on-demand, and pay-per-use manner. The rapid progress in virtualization technologies [26, 27] has opened a new paradigm in the cloud computing domain for sharing resources on-demand. Virtualization technologies allow dynamic provision of cloud resources as a result of user requests. Clouds can isolate the underlying resources and detailed implementation from end users.

Benefits of Cloud Computing

Cloud computing supports various domains, such as healthcare, scientific research, e-business, and e-commerce. It brings various benefits including:

- *Easy to access*: Cloud computing provides ubiquitous and heterogeneous services including IaaS, PaaS, and SaaS to various domains in an easy to access manner. This allows users to use resources anytime, anywhere around the world once they are connected to the Internet.
- *Elasticity*: Elasticity is a key part of Cloud computing and typically covers three facets: resource elasticity, cost elasticity, and quality elasticity [28].
- *Pay-as-you-go pricing model*: Cloud computing supports a pay-as-you-go pricing model that enables users to only pay for resources they use.
- *Focus on core business*: Cloud computing allows users to focus on their core businesses, i.e. users can leverage infrastructure resources from cloud providers instead of maintaining their own expensive hardware and the associated system administration costs.
- *SLA guarantees*: Cloud computing allows users to obtain resources and services provided by providers under professional maintenance with SLA guarantees for reliable services.

Service Level Agreements

SLAs represent agreements negotiated between end users and service providers to define the metrics, incentives, and penalties to be met during service delivery. They

reflect the constraints that a service should satisfy in a measurable way. SLAs are beneficial for both service users and providers as they provide specific defined service standards that must be met to deliver satisfactory services (to end users). As service providers are expected to fulfill user QoS requirements related to big data queries, admission control is needed to decide whether QoS can be met or not in order to determine whether the service request can/should be admitted for processing [6]. Customers should be able to compare the details of AaaS offered by different cloud providers, such as availability, cost, and processing times to decide which cloud service provider to choose.

A key element of an SLA is the service level objectives (SLOs), which are measurable characteristics of SLA. These can include response times, availability, throughput, etc. [29]. An SLO may include several QoS measurements with a percentage value specified over a give a measurement period, e.g. 95% of requests should be processed in less than 10 minutes. SLOs need to be fulfilled when delivering services to guarantee SLAs of user requests and avoid SLA violations that incur expensive penalty costs and decrease user satisfaction.

In most SLAs, penalties are defined based on the non-compliance of the agreements made to users when purchasing services. Given this, it is important for providers to meet/enforce QoS satisfaction to prevent SLA violations and continuously improve their offered services. Guaranteeing SLAs is essential to deliver services with high quality [161]. QoS requirements such as costs, response times and accuracy of results, may be required to be satisfied during resource allocation and service delivery. Otherwise, compensation may need to be made to users based on penalties defined as part of the SLAs.

In this thesis, we provide SLA guarantees on QoS requirements of queries including budget (the maximum cost to execute a query), deadline (the latest time to deliver query result), and accuracy (the confidence interval of query result), e.g. query results need to be returned within 1 hour within budget of 100 dollars and 95% accuracy. In order to avoid the risks of SLA violations and customer dissatisfaction, efficient and automatic admission control and resource scheduling mechanisms should be provisioned to guarantee SLAs.

Cloud Resources

Everything-as-a-Service (XaaS) [30, 31] envisages an emerging paradigm shaped by the advances of information technologies in cloud computing environments where all deliverable resources can be regarded as services. Cloud computing is a suitable platform for big data analytics by providing various resources at the infrastructure, platform and application level.

- *IaaS resources*: IaaS serves at the bottom layer of the cloud computing architecture and allocates on-demand virtualized resources including VMs [158], storage, networks, and containers.
- *PaaS resources*: PaaS offers an abstraction layer used to ease the development and deployment of applications for service providers who do not wish to manage the IaaS resources to support SaaS development. AaaS platforms are often delivered at the PaaS layer and broker IaaS and SaaS from third party providers and deliver AaaS services to various end user communities.
- *SaaS/Application resources*: At the SaaS layer, heterogeneous software services are provisioned to users. SaaS reduces the costs of users to purchase software licenses and enables users to lease applications in a pay-as-you-go model. SaaS resources benefit users of various domains and allows them to easily and directly utilize managed application services.

All of the above resources collectively ensure that the cloud computing paradigm is well-suited for big data analytics and the enablement of AaaS for various domains. Since BDAAAs are often targeted to specific domains, they are not well suited for occasional usage, individual users and/or small business. AaaS platforms need to serve on-demand analytic requests and deliver AaaS to multiple domains in an easy to use way and ideally at low costs to the end users.

2.1.3 Cloud-based Big Data AaaS Platforms

Big data platforms deliver AaaS to users in a wide range of application domains. Users typically submit analytic requests through a web interface to a given AaaS platform. Different analytic requests have different purposes and as such they can have different QoS requirements to fulfill. An AaaS platform should only admit analytic requests

whose QoS can be fulfilled with admission control based on pre-agreed SLAs related to guaranteed service delivery. Following this, the AaaS platform brokers cloud infrastructure resources and BDAA applications from third-party providers and delivers AaaS to users as consumable services in a pay-as-you-go model. An AaaS should reduce the burdens of users from various facets, such as dealing with BDAA licenses and maintaining and managing the underlying resources and dealing with their associated complexity.

AaaS platforms are located at the platform (PaaS) level and broker IaaS resources and SaaS resource from third-party providers. These are provisioned on-demand to users. AaaS platforms are expected to manage and schedule IaaS and SaaS resources in a cost-effective way and only allow queries with QoS requirements that can be met, i.e. within SLA guarantees.

There are an extensive number of works that focus on delivering specific services and provisioning data processing techniques related to AaaS platforms and enrich the AaaS can be provisioned. Chen et al. [32] tackled the research challenges in delivering continuous AaaS to analyze realtime events. Barga et al. [33] proposed Daytona, which offered scalable computation for large-scale data analytics focused on spreadsheet applications. Runsewe et al. [34] explored resource scaling from an application provider's perspective for time-bounded and time-unbounded big data streaming applications on the cloud. Chen et al. [35] studied crowdsourced live streaming systems to offer a cost-effective streaming service on the cloud. Kim et al. [36] proposed a cloud-based distributed multimedia streaming service based on Hadoop with suitable QoS. Li et al. [37] put forward Big Provision that delivered a provisioning framework aimed to effectively configure big data analytics systems based on performance evaluation and modeling of different data analytics approaches. Yang et al. [38] focused on architectural optimization and efficiency improvements from a machine learning perspective to better understand workloads and the environments. Ahmad et al. [39] studied interaction-aware scheduling for report generation workloads to optimize database performance. Yousfi et al. [40] studied big data AaaS solution for building graph-based social networks. Garcia et al. [41] introduced an SLA-aware system to manage the complete resource lifecycle of SLAs based on a framework for general cloud computing applications that could be dynamically adapted to correct potential

QoS violations. Gan et al. [154] studied a utility-driven data analytics algorithm for data pattern mining. Wan et al. [155] provided a multi-feature clustering algorithm for information extraction from big data.

There has been a number of works focused on leveraging Database-as-a-Service that provide database-level support for AaaS platforms. Relational cloud [42] explored technologies and challenges related to Database-as-a-Service in clouds. Popa et al. [43] proposed CryptDB to offer confidentiality and encrypted query processing. Nguyen et al. [44] supported query assurance verification for dynamic outsourced XML databases. Narasayya et al. [45] proposed SQLVM to provision multi-tenant relational Database-as-a-Service in multi-tenant Microsoft Azure SQL database. Wong et al. [46] proposed Thrifty to deliver a Database-as-a-Service with parallel query processing to achieve lower costs whilst meeting SLA guarantees on query results. Ramakrishnan et al. [47] proposed the Azure data lake store as a fully managed, elastic, scalable, and secure file system for big data analytics. Lee et al. [164] propose HANA as a distributed in-memory database system for transaction, session, and metadata management.

Several research works studied the delivery of specific AaaS solutions to users in diverse domains. Heterogeneous data analytics applications such as MapReduce applications [48, 49, 157], workflow applications [50, 51] and domain-specific applications such as healthcare and mobile applications are widely supported by various cloud service providers. Chen et al. [32] studied continuous analytics as a service that targets the processing of real-time events by delivering continuous analytics results for mobile applications. Zulkernine et al. [21] proposed CLAaaS as a big data analytic service provisioning platform targeting workflow applications. Lynn et al. [52] studied enterprise serverless cloud computing platforms that provide Function-as-a-Service. Khazaei et al. [53] proposed a framework for real-time and retrospective Health-Analytics-as-a-Service. Kumar et al. [54] explored SLA-based healthcare analytics service used to process healthcare data in the cloud for health condition predictions. Jindal et al. [55] focused on providing Healthcare-as-a-Service using fuzzy rule-based analytics with classification of big data generated on the cloud.

A number of works focused on providing data processing technologies to enable cloud platforms to provide effective and efficient AaaS solutions. Tordini et al. [56] proposed data splitting techniques to process next generation sequencing datasets for

complex workflows whilst keeping a high level of query accuracy. Zhang et al. [57] support normal distribution-based data splitting methods to process large medical data in cloud computing environments. Their work splits large datasets to obtain smaller representative data for processing thereby offering higher flexibility, less resource costs, and faster response times. Agarwal et al. [58] put forward BlinkDB to support approximate query processing on data samples from large datasets with bounded response times and errors for timely and reliable data analytics solutions. Hua et al. [59] proposed NEST, which offers ease-of-use and cost-effective approximate query service for the cloud. Their work enables fast approximate queries to enhance system scalability and improve the quality of service, especially when users are not aware of exact query inputs. NEST supports cloud applications in dealing with large volumes of data with increments of space utilization and fast query response times. These data processing technologies enable AaaS platforms to provide diverse big data analytics solutions for various application domains.

There have been various companies and providers that support data analytics services to users in various domains. Amazon offers data analytics services such as Elastic MapReduce [60] that enables easy and cost-effective processing of big data. Amazon charges users for the cloud resources and MapReduce services they offer. Google offers Big Query [61] as a fully managed, fast, economical database as a service for large-scale data analytics on the Google Cloud. IBM provides BigInsights [62] to provide Hadoop services that support security of an on-premises deployment on the cloud.

There are several widely used data analytics frameworks that support big data processing. Examples of these are Shark, Hive, Impala, and Tez/Stinger. These analytic frameworks are built on top of the Hadoop Distributed File System [63]. Hadoop is an open-source software framework developed by the Apache Software Foundation for reliable, scalable, and distributed computing [64]. Shark is a module in Spark, which allows interactive SQL on Hadoop systems [65]. Spark is an open-source computing framework that provides an easy-to-use programming model to leverage various applications, such as machine learning and graph processing applications [66]. Spark provides a fast compute engine based on the Hadoop file system. Hive provides ad hoc query processing on data warehouse infrastructures [67]. It allows large datasets to be

managed and facilitates SLAs using SQL-like language, named Hive SQL. Impala is a data analytic framework that enables multi-tenancy and low-latency SQL queries [68]. Impala supports parallel database technology that leverages Hadoop. Tez is a data-flow analytic framework that provides a powerful engine to execute both batch and interactive tasks in a flexible manner [69]. Tez brings the benefits of fast processing speed and data scalability. Tez allows users to conduct complex computations and analyze data using a customized execution architecture.

2.1.4 Resource Management for Cloud-Based Big Data AaaS Platforms

Resource management for AaaS platforms has become an important research topic to leverage big data analytics across a wide range of application domains. Admission control is used as a gateway to determine query admission to AaaS platforms. Maximization of query admission is a prerequisite for maximizing profits as well as satisfactory service delivery by rejecting any query whose QoS requirements cannot be fulfilled. Satisfactory AaaS fulfillment of QoS requirements for queries is essential to improve customer satisfaction. Resource scheduling should fulfill the functional requirements of big data analytics requests by BDAAAs with provisioned cloud resources.

Since big data analytics requests are typically resource intensive, they require a large amount of resources to be provisioned for subsequent execution. Scheduling mechanisms should therefore be scalable and elastic to acquire and release large amounts of resources as required at run time to optimize profits, reduce costs and guarantee SLAs. Scheduling solutions should also fulfill non-functional QoS requirements of the requests with SLA guarantees, e.g. budget, deadline, and accuracy constraints. Our proposed resource scheduling solutions serve the interest of AaaS platforms to maximize profits, increase market share, improve reputation by delivering timely, cost-effective and reliable AaaS with guaranteed SLAs.

Our research focuses on providing elastic and scalable cloud resource management algorithms to optimize the objectives of AaaS providers. Resource scheduling is responsible for assigning tasks to appropriate resources based on resource availability,

capacity, performance, etc. There are also often different objectives to fulfill, such as minimizing resource cost, minimizing response time, and satisfying QoS requirements of queries. In AaaS platforms, there can be a large number of online tasks coming into platforms at any time and each task may consist of sub-tasks with dependencies between them. Each task may require a specific service running on different resources for execution.

In this work we focus on providing efficient and effective admission control and resource scheduling algorithms to maximize profits while minimizing resource costs and query times with SLA guarantees. We study related research works compared to our work to ground the research and provide focus to the resource management solutions explored that form the contribution of this thesis.

2.2 Related Work

We categorize related works based on different scheduling objectives.

2.2.1 Cost and Time Optimization of Resource Management

Wen et al. [70] model a multi-objective optimization problem and propose a multi-objective workflow scheduling algorithm. They aim to minimize costs and processing times under privacy constraints. Their work does not focus on big data processing under budget, deadline, or accuracy constraints and they do not consider admission control or SLAs. Hu et al. [71] study multi-objective scheduling for scientific workflows in multi-Cloud environments with the aim to minimize workflow processing times and costs while satisfying reliability constraints. However, they do not consider budget, deadline, or accuracy constraints of requests or specifically target big data processing challenges under limited time or cost constraints. Kllapi et al. [72] address workflow scheduling problems dealing with time and money optimization. Chen et al. [165] proposed a real time scheduling algorithm to schedule big data workflows to optimize completion time and cost. However, they do not consider admission control, QoS requirements of tasks or SLA guarantees. Their work also does not consider big data processing under limited budgets and deadlines.

2.2.2 Cost Optimization of Resource Management

Mian et al. [74] schedule resources for data analytic workloads to determine cost-effective configurations through heuristic algorithms on public clouds. Their work schedules queries for one application and does not aim to support general data analytics scenarios. Moreover, their work trades-off SLAs for cheaper resource costs which is different from our SLA guarantee approach. We believe that SLA violations can significantly decrease user satisfaction. Therefore, SLA violations should be prevented for better service delivery and credibility of AaaS providers. Sun et al. [12] provision cost-effective solutions for multi-tenancy scenarios as part of their analytic framework. They propose an SLA customization mechanism to satisfy diverse QoS requirements of tenants. However, their work does not consider admission control. Thus, SLAs are at risk of violations. Moreover, they do not consider resource scheduling algorithms with SLA guarantees for profit enhancement of AaaS providers.

Garg et al. [75] proposed an admission control and resource scheduling algorithm in a cloud data centre that maximized the profit and resource utilization while ensuring the SLA requirements of users. This work considered deadline-constrained scheduling while we consider QoS constrained scheduling including budgets, deadlines and accuracy. Moreover, they manage resources for general cloud applications instead of big data analytics applications. Gu et al. [76] studied a cost minimization problem for big data processing in distributed cloud data centers. However, their work does not consider QoS requirements of query requests or supports SLA guarantees.

Zeng et al. [77] studied the cost minimization problem for big data processing in distributed clouds with focus on cost efficiency of batch data processing. Their work does not enable minimization of data analytics times and they do not consider QoS requirements of query requests or provide SLA guarantees. Calzarossa et al. [78] proposed a scheduling framework to minimize resource costs under deadline constraints for parallel data applications. They tackle uncertainties and the performance variability of cloud infrastructures and workloads. Their work does not consider admission control, automatic scheduling, or budget or accuracy constraints related to the data analysis tasks. They support a single objective optimization focused on costs, while we support multiple objective optimizations focused on costs and times. They also do not tackle big data processing challenges under tight deadlines and budget.

Xia et al. [79] studied a collaboration-aware and fairness-aware big data management problem to place generated data to distributed data centers to efficiently analyze the collected data and thereby save user costs associated with data storage and data processing. However, they do not consider QoS, SLAs, or AaaS delivery. Chen et al. [80] presented a utility model-based scheduling mechanism for cloud service providers to optimize profits against user satisfaction. However, they do not consider budgets, deadlines or accuracy as key QoS factors. Instead they trade-off QoS requirements of users which is different from our work as we aim to maximize user satisfaction by providing high quality AaaS satisfying QoS requirements. Gianniti et al. [81] provide a tool for cloud configurations with minimum cost of deployment for big data platforms that satisfies QoS constraints of tasks based on nonlinear programming. However, they do not consider SLA guarantees or tackle big data challenges related to deadlines or limited budgets.

Liu et al. [82] focus on online multi-workflow scheduling with the aim to minimize resource costs under deadline-constraints with random arrivals and uncertain task execution time in IaaS clouds. They do not consider time minimization, SLAs, accuracy or budget factors of tasks or task admission. Shao et al. [83] presented a market-oriented heuristic algorithm for scheduling parallel applications in a big data service platform with admission control to increase the platform operator's revenue. Their solution is not optimal as they do not ensure SLAs. This work also does not consider big data processing challenges under limited budgets or deadlines. Li et al. [84] presented a task scheduling method for MapReduce applications to improve the utilization of private clouds and reduce the cost of utilizing public clouds under budget and deadline constraints. Their work does not consider admission control, automatic provisioning, SLA guarantees, or accuracy factor of tasks.

Malekimajd et al. [85] formulated an optimization model for admission control and resource allocation for MapReduce jobs to minimize resource costs with deadline guarantees. However, they do not consider SLA guarantees on QoS parameters such as budgets or accuracy and they do not support automatic scheduling. Mao et al. [86] explored an auto-scaling mechanism for workflow scheduling to satisfy task deadlines whilst minimizing financial costs. However, their approach does not target big data analytics or consider budget or the accuracy requirements of requests. Ebrahimi et al.

[87] explored the scheduling of big data workflows on the cloud to minimize the execution cost of workflows under deadline constraints. Their approach focused on a single objective optimization (cost) while our solution considers multi-objective optimizations. They do not support big data processing under limited budgets or deadlines. They also do not consider budget or accuracy factors of tasks or SLAs.

Wang et al. [88] used integer programming for workflow scheduling in cloud environments for cost minimization and profit maximization under deadline constraints. Their solution does not consider task processing time minimization or tackling big data processing challenges under limited budgets or deadlines by applying effective data processing techniques. Lin et al. [89] proposed an online resource scheduling strategy for deadline constrained scientific workflows on hybrid clouds. However, they do not consider accuracy or budget requirements or tackle big data processing under tight budgets or deadlines. Zheng et al. [90] focused on scheduling of big data workflows to minimize costs while satisfying deadline constraints. However, they do not support admission control, automatic resource scheduling, or guarantee of SLAs. Lin et al. [91] propose a scheduling algorithm to minimize the execution cost of workflows while satisfying defined deadline constraints for workflows. However, they do not consider profit optimization, QoS requirements on budget or accuracy constraints or associated SLA guarantees.

Jrad et al. [92] explored healthcare workflows on multiple clouds by matching functional and non-functional SLA requirements with a utility-based genetic algorithm. They sought to benefit users through service quality and costs by reducing execution costs while fulfilling SLAs. Their work does not consider time minimization, admission control, automatic scheduling, big data features, or SLA guarantees associated with requests. Alrokayan et al. [93] proposed SLA-based algorithms to schedule cloud resources for MapReduce tasks to achieve cost-minimization under budget and deadline constraints. However, they do not consider admission control needed for SLA guarantees. Mattess et al. [94] proposed an algorithm to dynamically provision resources to meet soft deadlines of MapReduce tasks while minimizing budgets. However, their work does not consider admission control or SLA guarantees. Xiao et al. [95] proposed a cost-aware approach for big data processing in distributed datacentres to minimize costs of MapReduce applications while satisfying budget

constraints. However, they do not consider cost optimization or QoS requirements on deadlines and data query accuracies or SLA guarantees.

2.2.3 Time Optimization of Resource Management

Zhang et al. [96] explored multi-objective scheduling for many task workflows to find a near optimal solution to reduce the potential overheads. Their work neither considered SLA guarantees or profit optimization. Cao et al. [97] optimize performance of budget-constrained MapReduce tasks in cloud environments. They aimed to minimize the makespan of MapReduce tasks. Their work does not focus on profit optimization and they do not consider task admission or SLA guarantees. Tang et al. [98] proposed a self-adaptive scheduling algorithm to reduce start times of MapReduce tasks. However, their work does not consider profit optimization or SLA guarantees. Lavanya et al. [163] proposed SLA-based resource scheduling algorithms to reduce makespan and increase resource utilization. However, their work does not tackle big data processing challenges or consider admission control or profit optimization.

Kune et al. [99] proposed a scheduling model to address data intensive problems in big data clouds to minimize the turnaround time of jobs where the computing and data resources are decoupled. They proposed a genetic algorithm-based scheduler to find scheduling solutions. Their work does not optimize profit or other optimizations. Sun et al. [100] presented a multi-objective optimization problem for resource scheduling in cloud environments. Their work aimed to minimize the service latency with improved stability of task execution while we focus on profit optimization and data processing time minimization. However, they do not consider SLA guarantees of QoS requirements or target big data processing challenges.

2.2.4 Resource Management with Other Objectives

There are a number of works that aim to optimize scheduling performance in terms of various objectives including load balancing, energy-efficiency, performance/cost, efficiency, feasibility, utilization, success rates, throughput, and availability. We summarise the most relevant of these here.

Load Balancing

Fan et al. [101] proposed a dynamic resource scheduling method for load balancing among tasks and services for high performance computing applications in data centres. They do not consider QoS requirements of tasks, SLA guarantees, or big data processing challenges related to limited times or budgets and they do not focus on profit optimization scheduling. Tang et al. [102] proposed dynamic load-balance scheduling of latency-critical and computation-intensive applications in cloud environments. Their work does not consider profit optimization, SLA guarantees, or big data analytics under limited budgets or deadlines. Wang et al. [103] support data-aware scheduling techniques to optimize load balancing, however, they do not consider QoS, SLA guarantee, or cost optimization. Dou et al. [104] proposed a cloud resource co-allocation method for load balancing and scheduling of HPC applications in big data systems. They focus only on HPC applications. Moreover, they do not consider QoS parameters, cost optimization or automatic scheduling.

Energy Efficiency

Kaur et al. [105] put forward a data locality-aware energy-efficient scheduling approach for task scheduling and data placement of MapReduce tasks in cloud data centres. Their approach focuses on energy inefficiency minimization, deadline efficiency maximization, and data locality maximization instead of profit optimization for AaaS platforms. They also do not consider QoS of budgets or accuracy or task admission. Shao et al. [106] focus on efficient jobs scheduling approaches for MapReduce applications with the aim to reduce energy consumption while satisfying deadlines of user requests. Li et al. [162] study energy-efficient scheduling under customer satisfaction level constraint. Their work does not tackle big data processing challenges or guarantee SLAs.

Performance/Cost

Dai et al. [109] presented a multi-objective resource allocation approach for big data applications to obtain optimal deployment of VMs in clouds with the minimum cost and highest performance and availability. However, they do not consider admission control or tackle the time challenges related to big data processing. Zhou et al. [110]

provided a declarative optimization engine to provision cloud resources for workflows utilizing available GPU power for performance/cost optimizations with timely solutions. They neither consider QoS requirements nor SLA guarantees.

Feasibility

Anjos et al. [111] explored strategies for big data analytics with a data distribution model for resource management with improved performance for hybrid infrastructures on cloud and use of volunteer computing. They do not target profit optimization solutions and they do not consider SLAs of task processing or admission control.

Success Rates

Zhang et al. [112] focus on deadline constrained workflow scheduling to maximize resource utilization and success rates to meet task deadlines. They do not consider admission control to guarantee SLAs and they do not support automatic scheduling or consider budget or accuracy requirements. Arabnejad et al. [113] support budget and deadline aware resource scheduling algorithms with cost and time trade-offs to achieve higher success rates for scientific workflows, however, they do not tackle big data processing challenges under tight budget or deadline constraints, and they do not guarantee SLAs.

Throughput

Kollenstart et al. [114] target adaptive provisioning of heterogeneous cloud resources for big data processing to effectively reduce the under-utilization of resources in clusters to increase throughput and reduce lead times. Their approach targets CPU intensive applications suitable for data intensive tasks where data locality should be considered. They do not consider profit optimization of resource provisioning or SLA guarantees for task execution. Zhang et al. [115] presented solutions for evolutionary scheduling of dynamic multi-task workloads for big data analytics in cloud environments to achieve a higher overall scheduling performance for throughput. They rapidly process tasks in an iterative and evolutionary way by capturing the details of big-data workloads. However, their solutions are not optimal and the scheduling

solution at each period may be non-optimal. They do not consider profit optimization, SLAs, QoS requirements of deadline or accuracy or task admission.

Sfrent et al. [107] proposed resource scheduling for many task computing in big data platforms. They aimed to maximize the number of the jobs that could be processed, while we focus on proposing efficient profit optimization scheduling algorithms for AaaS platforms. Moreover, they do not consider QoS requirements or guarantee SLAs for scheduling. Malawski et al. [108] provided cost and deadline constrained provisioning algorithms for scientific workflow in IaaS cloud. Their objective was to maximize the number of workflows that could be completed given budget and deadline constraints. This work does not focus on big data processing challenges or profit optimization solutions. Their work also does not support big data processing under limited times and budgets.

Availability

Xiang et al. [116] proposed resource orchestration for multi-domains with focus on geo-distributed data analytics. The approach realized accurate, cross-domain resource availability and compute optimal resource allocation decisions to accurately represent the resource availability for a given set of jobs with minimal information exposure. However, they do not target profit optimization nor QoS of tasks, SLA guarantees for task execution, admission control, or tasks processing under limited budgets or deadlines.

2.3 Contextualizing the Thesis Contribution

The above related research works explored resource management from different perspectives and with different scenarios using various techniques and algorithms in cloud environments. However, none of these works support automatic and optimal admission control or cloud resource scheduling solutions that optimize the profit of AaaS providers and minimize times for queries while delivering cost-effective, reliable, and performant AaaS with SLA guarantees needed to tackle big data processing challenges under budget, deadlines, and accuracy constraints. Moreover, an extensive

number of related works either target a specific type of application or a specific user group that does not provide a general data analytics solution for AaaS platforms.

In order to fill these research gaps and support the continuous development of AaaS platforms, this thesis focuses on proposing effective and efficient admission control and cloud resource scheduling solutions for AaaS platforms to maximize profits and minimize query times with guaranteed SLAs on QoS requirements including budgets, deadlines, and the data query result accuracy. Our works support various applications and tasks with flexible structures and dependencies including single tasks, workflows, and more complex data analytics tasks such as optimization and machine learning algorithms to support AaaS delivery across multiple application domains. The proposed optimal admission control and resource scheduling solutions provide the foundation for AaaS platforms to maximize profits and minimize data analytics times whilst guaranteeing SLAs during AaaS delivery. This allows AaaS providers to enhance their reputation, improve user satisfaction, and ultimately increase their market share. The proposed resource management solutions address the proposed research problems in Section 1.2 are explored in the next five chapters.

3 SLA-Based Profit Enhancement for Resource Management

This chapter focuses on enhancing profits through efficient resource management solutions to deliver AaaS to users as consumable services with SLA guarantees. This chapter was previously published¹. It addresses Problem Scenario 1 as described in Chapter 1. The key contributions of this chapter are admission control and resource scheduling algorithms that enhance profits and deliver AaaS to users with minimized query times and SLA guarantees focused on budgets and deadlines. Specifically, we model and formulate a multi-objective problem targeted to automatic and scalable resource management solutions to tackle dynamic query requests, heterogeneity of cloud resources, different QoS requirements, potentially long data processing times, and expensive resource costs that typify big data processing.

The admission control algorithm estimates whether QoS requirements related to budgets and deadlines can be satisfied by any existing cloud resource configuration to make query admission decisions. The resource scheduling algorithm enhances profits to cloud providers by offering automatic solutions based on a two-phase policy. It schedules queries with maximized resource utilization plan in Phase 1 and subsequently scales resources up to provide cost minimized query execution plans in Phase 2. This two-phase scheduling enhances profits with elastic resource provisioning. It scales existing cloud resources up by leasing new resources based on dynamic resource demands and scales resources down by terminating over-provisioned resources. Experimental results show the proposed admission control and resource scheduling algorithms can effectively admit queries and offer significantly better results to creating higher profits and reduce resource costs compared to other state-of-the-art algorithms.

This chapter is derived from the following publication:

¹Y. Zhao, R. N. Calheiros, G. Gange, K. Ramamohanarao, and R. Buyya, “SLA-based resource scheduling for big data analytics as a service in cloud computing environments,” in *Proceedings of IEEE International Conference on Parallel Processing (ICPP)*, pp. 510–519, 2015.

SLA-Based Resource Scheduling for Big Data Analytics-as-a-Service in Cloud Computing Environments

**Yali Zhao, Rodrigo N. Calheiros, Graeme Gange, Kotagiri
Ramamohanarao, Rajkumar Buyya**

Department of Computing and Information Systems

The University of Melbourne, Australia

{yaliz1@student., rnc@, gkgange@, kotagiri@, rbuyya@}unimelb.edu.au

Abstract

Data analytics plays a significant role in gaining insight of big data that can benefit in decision making and problem solving for various application domains such as science, engineering, and commerce. Cloud computing is a suitable platform for Big Data Analytic Applications (BDAAAs) that can greatly reduce application cost by elastically provisioning resources based on user requirements and in a pay as you go model. BDAAAs are typically catered for specific domains and are usually expensive. Moreover, it is difficult to provision resources for BDAAAs with fluctuating resource requirements and reduce the resource cost. As a result, BDAAAs are mostly used by large enterprises. Therefore, it is necessary to have a general Analytics as a Service (AaaS) platform that can provision BDAAAs to users in various domains as consumable services in an easy to use way and at lower price. To support the AaaS platform, our research focuses on efficiently scheduling cloud resources for BDAAAs to satisfy Quality of Service (QoS) requirements of budget and deadline for data analytic requests and maximize profit for the AaaS platform. We propose an admission control and resource scheduling algorithm, which not only satisfies QoS requirements of requests as guaranteed in Service Level Agreements (SLAs), but also increases the profit for AaaS providers by offering a cost-effective resource scheduling solution. We propose the architecture and models for the AaaS platform and conduct experiments to evaluate the proposed algorithm. Results show the efficiency of the algorithm in SLA guarantee, profit enhancement, and cost saving.

3.1 Introduction

As we enter into the big data era, data analytics becomes critical in decision making and problem solving for various domains, such as science, engineering, commerce, and industry. Big data is referred to a massive volume of structured, unstructured, semi-structured, or real-time data, which is difficult to manage and process with traditional database techniques [4, 5]. Huge amount of data is generated from various sources, i.e., social media, sensors, websites, and mobile devices every day. The key enablers of big data are the increased capability of storage and computing resources enabled by recent advances in cloud computing and the increased availability of data and the ease of access to data supported by various companies like Amazon, Google, IBM, and Facebook. Cloud computing is a suitable platform for big data analytics by providing various resources, which are: (1) cloud infrastructures, such as Amazon EC2, which offer on-demand virtualized resources, i.e., Virtual Machines (VMs) and storages; (2) cloud platforms, such as Google App Engine, which manage underlying infrastructure resources and allow users to deploy and run applications; (3) application services, such as IBM Social Media Analytics [117], which are provided to users through web browser portals.

Analyzing big data to find potential insights in the data is essential for individuals, organizations, and governments to make better decisions, i.e., product trend prediction, business strategy making, and disaster prediction and management. However, big data analytics has several challenges. Obtaining analytic solutions is usually expensive as it requires large data storage systems to store voluminous data and considerable computing resources to analyze the data. Furthermore, licenses of Big Data Analytic Application (BDAA) are often expensive. As a result, big data analytics is mostly used by a small number of large enterprises [12]. Moreover, scheduling resources for BDAA requires expert knowledge. For example, in order to reduce the time of generating analytic solutions, BDAA should be deployed on various machines for parallel processing on large datasets, while in order to save energy and cost, resources should be scaled up and down for varied resource demands of BDAA. The lack of such expertise makes resource scheduling for BDAA difficult for many users. The above challenges illustrate the need for an Analytics as a Service (AaaS) platform, which aims at serving on-demand analytic requests and delivering AaaS to users as

consumable services in an easy to use way and at lower cost. The delivered AaaS should satisfy Quality of Service (QoS) requirements of requests with Service Level Agreement (SLA) guarantees. SLA is the agreement negotiated between service users and providers, which defines the metrics, expected QoS, and penalties during service delivery.

To support the AaaS platform, our research focuses on efficiently scheduling cloud resources for BDAAAs and provisioning BDAAAs to users as consumable services. We aim to serve the interests of AaaS providers to enhance profit by proposing cost-effective resource scheduling solutions, increase market share by accepting more requests, and improve reputation by delivering satisfactory services to users with SLA guarantees. BDAAAs that typically process read-only query requests on given datasets and leave out data consistency and security issues [74] are our targeted applications.

The key **contribution** of our work is an admission control and resource scheduling algorithm that admits queries based on QoS requirements, delivers analytic solutions to users with SLA guarantees, and provides a cost-effective resource scheduling solution to create higher profit for AaaS providers. To achieve this, we (1) propose the architecture and models of the AaaS platform; (2) formulate the resource scheduling problem based on mixed Integer Linear Programming (ILP) model [118] and propose the admission control and resource scheduling algorithm; and (3) conduct experiments to evaluate the performance of the algorithm in SLA guarantee, profit enhancement, and cost saving.

3.2 Architecture and Models

3.2.1 Architecture

We propose the architecture of the AaaS platform that provisions general AaaS to various domains of users as consumable services, as shown in Figure 3.1. The architecture is composed of the following components:

Admission controller makes decisions about whether to accept queries submitted by users. Admission controller first searches exhaustively in BDAA registry for requested BDAAAs and the cloud resource registry for all possible resource configurations. Then, it estimates query execution time and cost as basis to decide whether to accept or reject queries. Accepted queries are submitted to SLA manager.

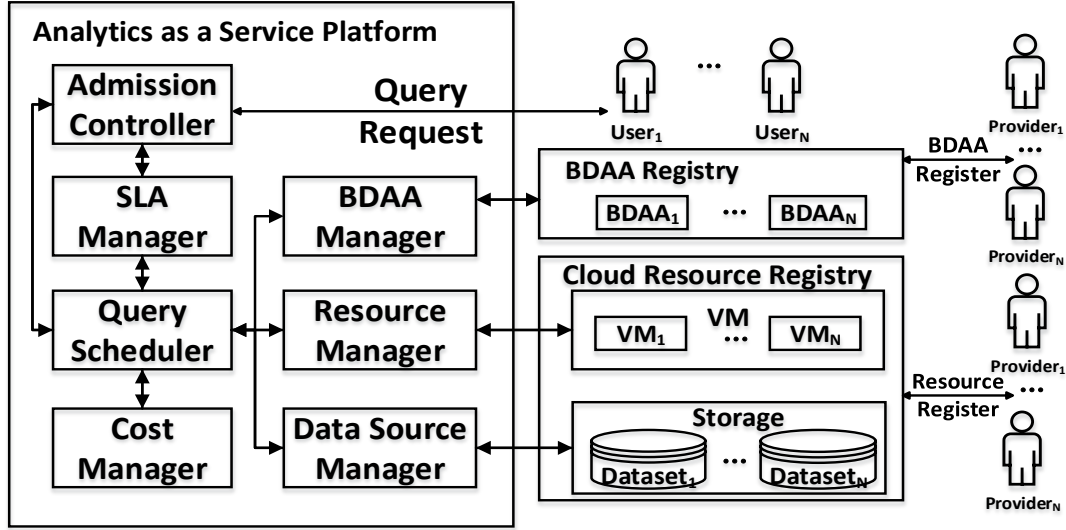


Figure 3.1: Architecture of the AaaS Platform.

SLA manager builds SLAs for accepted queries. SLA violations not only decrease user satisfaction but also generate additional penalty cost; thus, they should be avoided.

Query scheduler is the core component of the AaaS platform, which makes scheduling decisions and coordinates the other components. Scheduling decisions mainly contain: (a) **resource configuration**: the scheduler decides which type of cloud resource to use and how many to use for each resource type; (b) **VM control**: the scheduler sends VM controlling commands to the resource manager to control VMs, i.e., create VM, terminate VM, and migrate VM; (c) **BDAA selection**: the scheduler selects requested BDAA to execute a query; (d) **query execution sequence**: queries that request the same BDAA are submitted to the waiting queues of VMs running the BDAA (as queries contain deadline constraints, they are ordered for execution to avoid deadline violations); and (e) **query management**: the scheduler monitors and manages status of queries during their lifecycles. Query status can be one of submitted, accepted, rejected, waiting for execution, being executed, succeeded, and failed. Query status is monitored to allow the scheduler to take appropriate actions, i.e., send query with failed status to cost manager to generate penalty cost.

Cost manager manages all the cost occurred in the AaaS platform, i.e., query cost and resource cost. It also aims at providing pricing policies that can attract more users to enlarge market share and generate higher profit.

BDAA manager manages BDAAs provided by different BDAA providers and keeps up-to-date BDAA information.

Data source manager manages datasets that are to be processed. As big data has high volume, we move the compute to the data to save data transferring time and network cost.

Resource manager keeps a catalog of all available cloud resources from different providers and monitors status of VMs to allow the scheduler to take actions such as terminating idle VMs at the end of billing period to save cost.

3.2.2 Models

The models of the AaaS platform are shown as below:

Query request model: Query requests should contain the following information in query specifications: (a) QoS requirements, which contain budget and deadline to fulfill; (b) required resources to execute a query; (c) the requested BDAA; (d) data characteristics including data size, data type, and data location; (e) user who submits the query; and (f) query type, i.e., aggregation query and join query. Query requests can run on heterogeneous VMs and the query scheduler will determine the optimal resource configurations to execute queries.

Cloud resource model contains a set of datacenters and a matrix showing the network bandwidth between the datacenters. Each datacenter contains a set of hosts and data storages that pre-store datasets. Each host contains a set of VMs.

BDAA profile model contains resource configurations, data processing time, data size, data type, and costs for different queries. BDAA profiles are the basis for AaaS platform to estimate query time and cost to make admission and scheduling decisions. Obtaining the profiles requires expert knowledge. Thus, BDAA profiles are assumed to be provisioned by BDAA providers and are reliable.

Cost model contains sub-models of resource cost, penalty cost of SLA violations, query cost (query income) that charges users for using AaaS platform, and profit that is created for AaaS platform providers. Profit is the difference of query cost and the sum of resource cost and penalty cost. Query cost has different cost policies, which are: (a) cost based on urgency of deadline; (b) proportional cost, i.e., proportional to BDAA cost; and (c) a combination of (a) and (b). Resource cost contains BDAA cost and cloud resource cost. BDAA cost policies are: (a) fixed cost, i.e., annual contract; (b) usage

period-based cost, i.e., hourly basis; and (c) per request-based cost. Penalty cost policies are: (a) fixed cost; (b) delay dependent cost; and (c) proportional cost [119].

3.3 Admission Control and Resource Scheduling

We propose the admission control and resource scheduling algorithm that only admits queries whose QoS can be satisfied and schedules resources to execute queries with SLA guarantees to maximize the profit of AaaS providers. We adopt proportional query cost model, which calculates the query cost proportional to BDAA cost and fixed BDAA cost model, which is fixed cost under annual contract. We aim to make scheduling decision that can give 100% SLA guarantee. Thus, penalty cost of violating SLAs can be avoided if scheduling decision is well made. Since query cost and BDAA cost are fixed and penalty cost can be avoided, our goal of maximizing profit of the AaaS providers is to minimize the cloud resource cost.

3.3.1 Admission Control Algorithm

The admission controller makes query admission decisions. It first searches the BDAA registry to check whether a query requested BDAA exists. If the BDAA exists, admission controller obtains the BDAA profile. Based on the profile, expected execution time and query cost for different resource configurations are calculated. The expected query cost is calculated based on the adopted query cost model. The expected finish time is the sum of estimated execution time, specified timeout (the maximum time the scheduling algorithm can run), VM creation time (the time to create new VM when needed), submission time (the time when the query is submitted), and waiting time (the time till the query is scheduled). Admission controller makes query acceptance decision if estimated finish time and cost of queries can satisfy QoS requirements of queries using any resource configuration. Afterwards, SLA manager builds SLAs for accepted queries.

3.3.2 Scheduling Algorithms

After the admission controller accepts queries, the query scheduler makes the scheduling decision for each BDAA. We develop scheduling algorithms that support different scheduling scenarios, which are periodic scheduling that schedules queries for each Scheduling Interval (SI) and non-periodic (real time) scheduling that schedules queries whenever they arrive. For periodic scheduling, we set different SI parameters to study

how SI can influence the performance of the scheduling algorithms. We detail the study of algorithm performance for different scheduling scenarios in performance evaluation section.

We aim at scheduling and provisioning resources for BDAAAs that can maximize profit for AaaS providers by minimizing resource cost and guarantee SLAs of queries on deadline and budget. The scheduling algorithms adopt a two-phase scheduling policy that provision resources in a scalable and elastic way. They scale resources down by releasing resources when the provisioned resource capacity is more than required to reduce cost and scales resources up by leasing new resources when provisioned resources do not have sufficient capacity to execute queries to avoid SLA violations.

ILP scheduling algorithm

We model and formulate the two-phase resource scheduling problem based on ILP model [118]. Phase 1 aims at minimizing resource cost by scheduling queries to existing VMs to maximize VM utilization while satisfying SLAs of queries. If queries are not successfully scheduled, they will be handled by Phase 2 for further scheduling, which aims to minimize the cost of creating VMs to execute queries with SLA guarantees.

Phase 1 of the scheduling problem is formulated as a multi-objective ILP problem [120]. Minimizing cost of resources for query execution under budget and deadline constraints is an NP-complete decision problem that can be polynomially reduced to an ILP problem. The optimization objective of Phase 1 scheduling is minimizing resource cost. To achieve this, we have three individual objectives to fulfill, which are: **Objective A** aims to maximize resource utilization by assigning queries to existing VMs using maximized VM capacity, as shown in (1), where n denotes the number of created VMs; m indicates the number of queries to be scheduled; L_i is the required cloud computing resource to execute the query such as single VM or a cluster of VMs; and x_{ij} denotes whether $Query_i$ is assigned to VM_j (if $Query_i$ is assigned to VM_j , $x_{ij} = 1$; otherwise, $x_{ij} = 0$). **Objective B** aims to save resource cost by executing queries on VMs with lower cost first to reduce load on VMs with higher cost to allow VM termination, as shown in (2), where C_j is the cost of VM_j (modeled on an hourly basis) and y_i denotes whether VM_j is to be terminated (if VM_j is to be terminated, $y_j = 0$; otherwise, $y_j = 1$). **Objective C** aims to reduce VM runtime for cost saving purpose and to reduce query response time for better performance by executing queries at the earliest time, as shown in (3), where s_i is the starting time of $Query_i$.

$$A = \max(\sum_{j=1}^n \sum_{i=1}^m (L_i * x_{ij})) \quad (1)$$

$$B = \max(-\sum_{j=1}^n (C_j * y_j)) \quad (2)$$

$$C = \max(-\sum_{i=1}^m s_i) \quad (3)$$

We formulate the overall optimization Objective D as a combination of three individual objectives, as shown in (4), subjects to constraints (5)-(16). The combination leads to a standard lexicographic optimization problem [121]. The importance of the above objectives in contributing to the optimization problem is $A > B > C$. Coefficient F_0 and F_1 are given accordingly to Objective A and Objective B to guarantee the aggregated problem as maximization of all individual objective functions is consistent to the original problem, as shown in (17) and (18). The symbols of the ILP model are defined in Table 3.1.

Table 3.1: Definition of Symbols

Symbol	Definition
i	A specific query, $Query_i$
k	A specific query, $Query_k$
j	A specific VM, VM_j
m	The set of queries
n	The set of VMs
L_i	The required resources of a query
x_{ij} / x_{kj}	It indicates whether $Query_i/Query_k$ is assigned to VM_j
C_j	The cost of VM_j
y_j	It indicates whether a VM_j is to be terminated
s_i / s_k	The starting execution time of $Query_i/Query_k$
CP_j	The available capacity of VM_j
b_{ik} / b_{ki}	It indicates whether $Query_i$ executes before $Query_k$
e_{ij}	The execution time of $Query_i$ on VM_j
D_i	The deadline of $Query_i$
B_i	The budget of $Query_i$
C_{ij}	The cost of executing $Query_i$ on VM_j
z_j	It indicates whether a VM_j is to be created

Objective:

$$D = \max (F_0 * \sum_{j=1}^n \sum_{i=1}^m (L_i * x_{ij}) - F_1 * \sum_{j=1}^n (C_j * y_j) - \sum_{i=1}^m s_i) \quad (4)$$

Subject to:

$$\sum_{i=1}^m (L_i * x_{ij}) \leq CP_j, \forall i \in m, j \in n \quad (5)$$

$$x_{ij} = \{0|1\}, \forall i \in m; j \in n \quad (6)$$

$$b_{ik} + b_{ki} \leq 1, \forall i, k \in m \quad (7)$$

$$b_{ik} = \{0|1\}, \forall i, k \in m \quad (8)$$

$$b_{ik} + b_{ki} - x_{ij} - x_{kj} \geq -1, \forall i, k \in m; j \in n \quad (9)$$

$$s_i - s_k + F_2 * b_{ik} \leq F_2 - e_{ij}, \forall i, k \in m; j \in n \quad (10)$$

$$s_i + F_3 * x_{ij} \leq F_3 + D_i - e_{ij}, \forall i \in m, j \in n \quad (11)$$

$$C_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (12)$$

$$\sum_{j=1}^n x_{ij} \leq 1, \forall i \in m, j \in n \quad (13)$$

$$y_j \geq x_{ij}, \forall i \in m, j \in n \quad (14)$$

$$y_j \geq y_{j+1}, \forall j \in n \quad (15)$$

$$y_j = \{0|1\}, \forall j \in n \quad (16)$$

VM capacity constraints: as shown in (5), where x_{ij} is a binary variable indicating whether $Query_i$ is scheduled to VM_j , as shown in (6), CP_j denotes the available capacity of VM_j before the maximum deadline of the m queries. Constraint (5) ensures that the overall resources required by queries do not exceed the available capacity of VM_j .

Query deadline constraints: as shown in (7) to (11), where b_{ik} is a binary variable as shown in (8), which denotes whether $Query_i$ is executed before $Query_k$, if it is, $b_{ik} = 1$; otherwise, $b_{ik} = 0$. Constraint (7) is used to ensure the unique execution order of $Query_i$ and $Query_k$ by allowing maximum one of b_{ik} and b_{ki} to be 1. Constraint (9) restricts that $Query_i$ must be executed either before $Query_k$ or after $Query_k$ when they are executed on the same VM_j , which is derived from non-linear constraint (19). Constraint (19) ensures that if $Query_i$ and $Query_k$ are scheduled on VM_j , either $b_{ik} = 1, b_{ki} = 0$ ($Query_i$ executes before $Query_k$) or $b_{ik} = 0, b_{ki} = 1$ ($Query_i$ executes after $Query_k$). Constraint (10) is derived from non-linear constraint (20), which is used to restrict if $b_{ik} = 1$, $Query_i$ should finish before $Query_k$ starts execution, where e_{ij} denotes the execution time of $Query_i$ on VM_j and F_2 is a sufficient large constant satisfying (21). Constraint (11) is used to ensure that if $Query_i$ starts execution on VM_j

at s_i , it should finish before its deadline D_i , derived from non-linear constraint (22), where F_3 is a sufficient large constant satisfying (23).

Query budget constraints: as shown in (12), where C_{ij} is the cost of executing $Query_i$ on VM_j . Constraint (12) is used to guarantee that execution cost of $Query_i$ on VM_j do not exceed its budget B_i .

Query scheduling times constraints: as shown in (13), which indicates that query can be either scheduled in Phase 1 ($\sum_{j=1}^n x_{ij} = 1$) or not ($\sum_{j=1}^n x_{ij} = 0$). If there is any query not scheduled in Phase 1, it is to be scheduled in Phase 2.

VM termination constraints are shown in (14) to (16), where y_j is a binary variable denotes whether VM_j is to be terminated, as shown in (16). Constraint (14) is used to guarantee when VM_j is to be terminated, a query should not be assigned to it, which constrains when $y_j = 0$, x_{ij} must be 0 ($Query_i$ should not be assigned to VM_j); when $y_j = 1$, x_{ij} can be 0 or 1 ($Query_i$ can either be assigned to VM_j or not). Constraint (15) is used to provide the priority of utilizing created VMs. Created VMs are input to ILP in a cost ascending list. Thus, (15) can allow the scheduler to use VMs with cheaper cost first, which allows load decreasing on VMs with higher cost to terminate VMs when they become idle for cost saving. If VMs have the same price, the query scheduler first uses VMs in front of the VM list. Moreover, the scheduler checks periodically whether any VM is idle. If there is an idle VM, the scheduler further checks whether the VM is reaching the end of its billing period. If it is, the scheduler releases the VM to serve the objective of cost minimization.

$$F_0 = \max(B + C) - \min(B + C) + 1 \quad (17)$$

$$F_1 = \max(C) - \min(C) + 1 \quad (18)$$

$$\begin{aligned} x_{ij} = 1 & \text{ either } \begin{cases} b_{ik} = 1, b_{ki} = 0 \\ b_{ik} = 0, b_{ki} = 1 \end{cases}, \forall i, k \in m, j \in n \\ x_{kj} = 1 & \implies \begin{cases} b_{ik} = 1, b_{ki} = 0 \\ b_{ik} = 0, b_{ki} = 1 \end{cases} \end{aligned} \quad (19)$$

$$b_{ik} = 1 \Rightarrow s_i + e_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (20)$$

$$F_2 \geq \max(s_i + e_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (21)$$

$$s_i * x_{ij} \leq D_i - e_{ij}, \forall i \in m, j \in n \quad (22)$$

$$F_3 \geq \max(s_i + e_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (23)$$

Phase 2 of scheduling starts to scale resources up by creating VMs to execute queries that are not scheduled in Phase 1. The objective is to minimize VM cost with guaranteed SLAs in query execution, as shown in (24). Objective E is subjected to all constraints the same as in Phase 1, except constraint (13). Changed constraint is shown in (25),

which is used to restrict that $Query_i$ has to be scheduled to one of the newly created VMs for execution to avoid SLA violation. We use a greedy algorithm to decide the initial number of VMs of each VM type to input to Phase 2 of ILP algorithm. This ensures the capacity of input VMs is close to the required capacity of the optimal solution to avoid ILP taking long time to find a solution, which greatly reduces the algorithm running time of ILP. Given that input VMs obtained from the greedy algorithm have sufficient capacity to execute queries, to serve the cost minimization objective, ILP decides whether to create each of input VMs to execute queries, indicated by z_j . If $z_j = 1$, ILP creates input VM_j ; otherwise, ILP does not create the VM. After Phase 1 and Phase 2, all queries are successfully scheduled to VMs for execution.

$$E = \min \sum_{j=1}^n (C_j * z_j) \quad (24)$$

$$\sum_{j=1}^n x_{ij} = 1, \forall i \in m, j \in n \quad (25)$$

Based on the above formulation of the scheduling problem, we build the ILP scheduling algorithm. ILP algorithm first gets all accepted queries for each requested BDAA. In Phase 1, ILP schedules queries to existing VMs running the BDAA to maximize VM utilization and execute queries with SLA guarantees. Meanwhile, ILP tries to scale down resources to save resource cost and executes queries at the earliest time to achieve better performance. Afterwards, ILP takes scheduling actions based on returned scheduling solution. After Phase 1, if there are queries not scheduled, Phase 2 generates scheduling solutions by scaling up VM resources to execute queries with minimized resource cost and SLA guarantees. Then, ILP takes actions based on the returned scheduling solutions of Phase 2.

Adaptive Greedy Search (AGS) scheduling algorithm

We propose the AGS scheduling algorithm, inspired by previous works [74, 119], which aims to find a cost-effective VM configuration to execute queries while satisfying SLAs of queries by applying a greedy search heuristic. For each BDAA, in Phase 1 of scheduling, the AGS algorithm first gets queries requesting the BDAA and the created VMs running the requested BDAA. If the BDAA is requested for the first time, one initial VM is created. AGS first tries to use all existing VMs to execute accepted queries, as shown in Lines 1-9 of the pseudo code below. AGS schedules all queries based on the urgency of deadline, which is represented by Scheduling Delay (SD). SD is the difference between deadline and expected finish time of the query. AGS first sorts queries based on SD in an ascending order for scheduling; then, AGS tries to assign each

query to a VM that can satisfy its SLAs and gives it the Earliest Starting Time (EST).

We name this scheduling method as the SD-based method.

AGS scheduling algorithm

Input: accepted queries and current VM configuration

Output: query scheduling solution and new VM configuration

```

1: for each BDAA  $\in$  requested BDAA list
2:   queries  $\leftarrow$  get accepted query list of BDAA
3:   VMs  $\leftarrow$  get running VM list of BDAA
4:   if size(queries) > 0
5:     create initial VM for BDAA if it is firstly requested
6:     queries  $\leftarrow$  sort queries based on SD
7:     for each query  $\in$  queries
8:       do Phase 1 schedule to assign query to VMEST that can satisfy its SLAs and
       update EST of VMEST
9:     end for
10:    queries  $\leftarrow$  get queries not scheduled if they exist
11:    if queries contain at least one query do Phase 2 schedule
12:      CMs  $\leftarrow$  get all possible configuration modifications
13:      continueSEARCH  $\leftarrow$  true
14:      iterationN  $\leftarrow$  0
15:      iteration2N  $\leftarrow$  0
16:      Cost(AGS)cheapest  $\leftarrow$  a constant which is sufficient large
17:      while continueSearch is true or iteration2N > 0
18:        iterationN++
19:        iteration2N--
20:        for iterator  $\leftarrow$  CMs; iterator has next CM
21:          CM1  $\leftarrow$  get next CM from iterator
22:          while iterator has next CM
23:            CM2  $\leftarrow$  get next CM from iterator
24:            C1  $\leftarrow$  get new VM configuration by applying CM1
25:            C2  $\leftarrow$  get new VM configuration by applying CM2
26:            Cost(AGS)1  $\leftarrow$  calculate cost of C1 based on SD
27:            Cost(AGS)2  $\leftarrow$  calculate cost of C2 based on SD
28:            C  $\leftarrow$  choose from C1 and C2 for cheaper Cost(AGS)
29:            Cost(AGS)C  $\leftarrow$  cheaper Cost(AGS)
30:          end while
31:        end for
32:        if Cost(AGS)C < Cost(AGS)CHEAPEST,
33:          CCHEAPEST  $\leftarrow$  C;
34:          Cost(AGS)CHEAPEST  $\leftarrow$  Cost(AGS)C
35:        end if

```

```

36:   else
37:        $\text{continue}_{\text{SEARCH}} \leftarrow \text{false}$ 
38:        $\text{iteration}_{2N} \leftarrow 2 * \text{iteration}_N$ 
39:   end else
40: end while
41:   adopt  $C_{\text{CHEAPEST}}$  and take according scheduling actions
42: end if
43: end if
44: end for

```

After Phase 1, if there still are some queries not scheduled due to QoS constraints on deadline and budget, AGS starts Phase 2 of scheduling to find a cost-effective VM configuration to create new VMs for query execution, as shown in Lines 10-40. The set of all possible VM configurations for a requested BDAA is represented by a directed acyclic graph, $\text{Configurations} = (N, E)$, where N is a set of nodes and E is a set of edges. A node is a possible VM configuration C_i . An edge (C_i, C_{i+1}) indicates VM configuration C_{i+1} can be obtained from VM configuration C_i by applying a Configuration Modification (CM). The number of VM types available in the AaaS platform determines the number of possible CM s. The CM s are adding the cheapest VM, adding a more expensive VM, adding a next more expensive VM if it exists, till adding the most expensive VM.

Firstly, AGS gets all possible CM s and iteratively apply each CM on current resource configuration to generate a new configuration. Then, the cost manager calculates $\text{Cost}(AGS)_1$ and $\text{Cost}(AGS)_2$ of resource configurations C_1 and C_2 in the following way: for each configuration, the scheduler applies the SD-based method to schedule queries. Then the cost manager calculates the cost of the scheduling solution, which is the sum of resource cost and penalty cost. The penalty cost is set to a sufficiently high value that generates high $\text{Cost}(AGS)$ for SLA violations. This causes AGS selecting a cheaper resource configuration without violating SLAs of queries. In each search iteration, AGS selects the configuration C that gives cheapest $\text{Cost}(AGS)_c$. Then, AGS compares $\text{Cost}(AGS)_c$ with the cheapest $\text{Cost}(AGS)_{\text{cheapest}}$ generated thus far. If the $\text{Cost}(AGS)_c$ is cheaper, the cheapest configuration C_{cheapest} is set as C while the cheapest cost $\text{Cost}(AGS)_{\text{cheapest}}$ is set as $\text{Cost}(AGS)_c$. AGS scheduling algorithm continues these steps until it finds the first local optimal configuration using N iterations (iteration_N). Afterwards, it continues exploring for another $2N$ iterations

($iteration_{2N}$) to see whether it can find another local optimal configuration that can give a cheaper $Cost(AGS)$. The cheapest configuration found after these $3N$ iterations is chosen. Finally, the scheduling solutions are adopted to execute queries, as shown in Line 41.

Adaptive ILP (AILP) scheduling algorithm

ILP can find the optimal solution for the scheduling problem. However, as an optimization algorithm, the Algorithm Running Time (ART) of ILP increases quickly as the input size grows, which might lead to deadline violations when ILP takes a long time to find scheduling solution. In case that an unexpected large number of queries arrive, to avoid the risk of violating deadline in such situation, we integrate AGS and ILP to form a new scheduling algorithm, which is AILP.

AILP works in the following way: it first utilizes ILP to make scheduling decision and specifies a timeout for ILP to give scheduling solution to limit the ART of ILP. When timeout is reached, if feasible integer linear programming solution is found (which may not be the optimal one), ILP returns the suboptimal solution. If no feasible solution is found, ILP only returns the timeout. After the scheduling of ILP, if there is any query that is not successfully scheduled, AILP utilizes AGS as the alternative scheduling algorithm to make scheduling decision to avoid SLA violations. The reason that AILP utilizes ILP at the first place to give scheduling decisions is to maximize profit and guarantee SLAs. Only if timeout of ILP is reached, to avoid deadline violations that can lead to user dissatisfaction and additional penalty cost, AILP utilizes AGS as an alternative scheduling algorithm. In this way, AILP serves as the best solution to make scheduling decisions for the purposes of profit maximization and SLA guarantee while overcoming the ART limitation of ILP, which is utilized as the scheduling algorithm for the AaaS platform.

3.4 Performance Evaluation

To evaluate the proposed admission control and scheduling algorithm, we build the framework of AaaS in CloudSim [122], which is a discrete event simulator. CloudSim enables repeatable and controllable experiments and offers comprehensive environment for resource scheduling and provisioning studies. The resource scheduler utilizes `lp_solve 5.5` [127] as the integer linear programming solver. We modeled cloud resources based on Amazon EC2 instances. We conduct experiments on the AaaS

framework to evaluate the effectiveness and efficiency of the proposed admission control and resource scheduling algorithm for SLA guarantee, profit enhancement, and cost saving. ILP and AGS are utilized as baseline algorithms for the evaluation of AILP scheduling algorithm.

3.4.1 Resource Configuration

We simulate a datacenter that consists of 500 physical nodes. Each node has 50 CPU cores, 100GB memory, 10TB storage, and 10GB/s network bandwidth. We simulate five types of memory optimized VMs, which are based on Amazon EC2 VM model and managed by the resource manager. The VMs are r3.large, r3.xlarge, r3.2xlarge, r3.4xlarge, and r3.8xlarge, as shown in Table 3.2 [123]. VMs in Amazon EC2 are charged by an hourly basis with unit of dollar/hour. The unit for memory is GiB while for storage is GB.

Table 3.2: VM Configuration

Type	vCPU	ECU	Memory	Storage	Cost
r3.large	2	6.5	15	32 SSD	0.175
r3.xlarge	4	13	30.5	80 SSD	0.350
r3.2xlarge	8	26	61	160 SSD	0.700
r3.4xlarge	16	52	122	320 SSD	1.400
r3.8xlarge	32	104	244	640 SSD	2.800

3.4.2 Workload

The workload generated in this study is based on the Big Data Benchmark [124], which runs query applications using different data analytic frameworks on large datasets with given data size, data type, and data location. The benchmark uses Amazon EC2 VMs and details query processing time and the VM configurations. Based on the information, resource requirements of queries requesting different BDAAs under different resource configurations are modeled in CloudSim.

Each query request contains the following information: **Query submission time** is generated using a Poisson process with 1-minute mean Poisson arrival interval. **Query**

class contains 4 types, which are: scan query, aggregation query, join query, and User Defined Function (UDF) query. **BDAA**s: in our experiment, 4 types of BDAA are considered, which are built on Impala (disk) (BDAA₁), Shark (disk) (BDAA₂), Hive (BDAA₃), and Tez (BDAA₄). **Query resource** is modelled based on the resource requirements of queries. As the performance of queries vary, we model 10% performance variation by introducing a variation coefficient [125], which is generated from a Uniform Distribution with upper bound 1.1 and lower bound 0.9. **Query user**: we simulate 50 users who submit queries and request for BDAA. **Query deadline** is generated as the benchmark does not contain QoS requirements on deadline (the same for budget). Two types of deadline are considered, tight deadline and loose deadline. Tight deadline is generated using a Normal Distribution (3, 1.4), where 3 represents that the mean deadline of a query is 3 times of its processing time and 1.4 represents the standard deviation. Loose deadline is generated similarly using a Normal Distribution (8, 3) [126]. **Query budget** contains tight budget and loose budget. Tight budget is generated using a Normal Distribution (3, 1.4) while loose budget is generated using a Normal Distribution (8, 3).

3.4.3 Result Analysis

We generate an approximate 7 hours query workload that contains 400 queries. The requested BDAA by queries are managed by the BDAA manager. Queries whose QoS requirements on deadline and budget can be satisfied are admitted by admission controller. To avoid deadline violation and ensure the performance of queries, the number of queries assigned to VMs by the query scheduler is less than the number of VM cores to avoid time sharing between queries as time sharing may cause deadline violations. To evaluate the effectiveness and efficiency of the admission and scheduling algorithm, we conduct the following studies.

Admission control and SLA guarantee

We conduct the real time scheduling and periodic scheduling by varying SI from 10 to 60 (unit: minute) to evaluate the effectiveness of the admission control algorithm. The results are shown in Table 3.3, where SQN is the submitted query number, AQN is the accepted query number, and SEN is the successfully executed query number.

We calculate the acceptance rate based on the obtained results. For real time scheduling, the rate is 84.0%, while for periodic scheduling of SI from 10 to 60, the rate

is 79.3 %, 74.8 %, 71.8 %, 68.5 %, 65.3 %, and 63. 0%. As the SI increases, the acceptance rate of queries decreases due to deadline constraints. Short SI is recommended as more queries can be admitted to increase user satisfaction and enlarge market share. We can see that all accepted queries are successfully executed with SLA guarantees, which helps to increase the reputation of AaaS providers. Obtained results show the effectiveness of the admission control algorithm in query admission.

Table 3.3: Query Number Information

Number	Real Time	Periodic					
		10	20	30	40	50	60
SQN	400	400	400	400	400	400	400
AQN	336	317	299	287	274	261	252
SEN	336	317	299	287	274	261	252

Cost saving and profit enhancement

In this study, we evaluate the efficiency of AILP, ILP, and AGS scheduling algorithms in cost saving and profit enhancement. AILP is an integration of AGS and ILP, it first utilizes ILP to find scheduling solution and only switches to AGS when no feasible solution is returned by ILP within specified scheduling timeout to avoid potential deadline violations. When timeout of AILP is reached, we record the data to show the contribution of ILP and AGS in generating the scheduling solution of AILP. For periodic scheduling using ILP, we only record the scheduling solutions that are given within the SI. As scheduling solutions exceeding the SIs are not applicable, from this paragraph on, we focus on algorithm comparison between AILP and AGS for simplicity. Obtained results are shown in Figure 3.2. The reason Figure 3.2 does not include resource costs of ILP for SI>20 is that ILP does not return feasible solution for SI>20 and hence the resource costs are not available. All resource costs for AGS, AILP, ILP are plotted in Figure 3.2 where available. We note that for real time scheduling and periodic scheduling with SI=10 and SI=20, ILP algorithm can schedule queries within the specified timeout. This indicates that scheduling decisions of AILP for these SIs are optimal and are given by ILP. For SI=30 and SI=40, in some SIs, scheduling timeout of ILP is reached, which indicates the solution given by AILP might be sub-optimal. For

SI=50 and SI=60, AGS is utilized along with ILP to contribute to the scheduling solution of AILP. Based on the obtained results, we can know that, for real time scheduling, the resource cost of AILP is 7.3% less than AGS, while for periodical scheduling with SI from 10 to 60, resource cost generated by AILP is 11.3%, 9.3%, 4.8%, 4.4%, 5.4%, and 4.3% less than AGS.

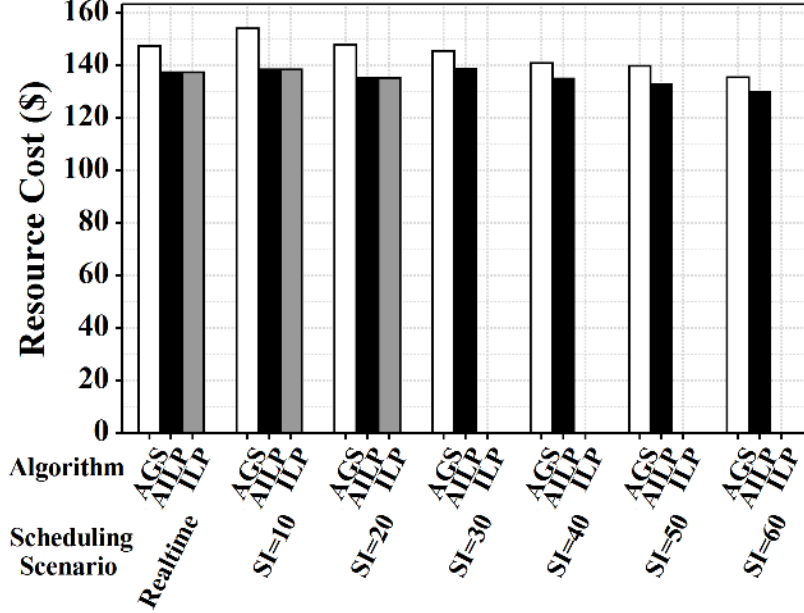


Figure 3.2: Resource Cost of AGS, AILP, and ILP.

The resource configuration to execute the query workload is shown in Table 3.4. Results show that, for all scheduling scenarios, AILP can efficiently reduce the numbers of VMs that are used. The observation is consistent with the Objective B and Objective E, as shown in (2) and (24), and constraint (15). ILP uses cheaper VMs at first if they are available; then, for the same VM type, ILP first uses VMs in the front of the VM queue. This leads to cheaper resource cost and higher resource utilization, which also leads to less VM creations. Creating less VMs is beneficial as it takes several minutes to boot VMs and make VMs online. We use 97 seconds as VM configuration time in our experiment [128]. Creating more VMs will cause more delay to serve requests, which may cause deadline violations and lead to higher resource cost. Thus, less VM creations can better benefit our objective of SLA guaranteeing and cost saving. Results show that the VM types utilized by AILP and AGS are r3.large and r3.xlarge while other types of r3.2xlarge, r3.4xlarge, and r3.8xlarge are not utilized. We find the reason that,

indicated by Table 3.2, there is no pricing advantages to use VMs with larger capacity as the capacity of VM increases, the price increases proportionally. Since VMs is not always fully utilized and creating VMs with higher capacity leads to higher resource cost, expensive VMs are not given the priority for utilization.

Table 3.4: Resource Configuration

Scheduling Scenario		AGS	AILP
<i>Real Time</i>		58 * r3.large	23 * r3.large
<i>Periodic</i>	<i>SI=10</i>	48 * r3.large	23 * r3.large
	<i>SI=20</i>	27 * r3.large	22 * r3.large
	<i>SI=30</i>	32 * r3.large	22 * r3.large
	<i>SI=40</i>	28 * r3.large, 2 * r3.xlarge	22 * r3.large
	<i>SI=50</i>	28 * r3.large	17 * r3.large, 2 * r3.xlarge
	<i>SI=60</i>	21 * r3.large, 4 * r3.xlarge	16 * r3.large, 2 * r3.xlarge

Profit gained by AILP and AGS is shown in Figure 3.3. We notice that for both real time scheduling and periodic scheduling, AILP performs significantly better than AGS in profit enhancement. For real time scheduling, profit gained by AILP is 11.4% more than AGS. For periodic scheduling of SI from 10 to 60, profit gained by AILP is 19.8%, 15.2%, 7.9%, 6.7%, 8.2%, and 6.1% more than AGS.

For real time scheduling, profit created by AILP and AGS is relatively high as more queries are admitted that leads to higher query income. The profit decreases when the real time scheduling is switched to the periodic scheduling with SI=10 as more queries are rejected, which decreases the query income. With the continuous increase of SI, as more queries are scheduled during each SI, AGS and AILP can make better scheduling decisions to provision resources and schedule queries to VMs, which increases the profit. For AGS with SI from 10 to 60, we notice the overall increasing trend of profit is relatively obvious comparing to AILP. For AILP, we notice that profit increases when SI increases from 10 to 20. However, when SI increases from 20 to 30, the profit drops as AILP returns sub-optimal solutions in some SIs when timeout is reached. We further

notice that when SI increases from 30 to 40, the profit starts increasing again, which indicates that AILP can give higher cost saving to create higher profit even though the query income is decreased when more queries are rejected. When SI of AILP increases from 40 to 50 and from 50 to 60, the profit keeps increasing as more queries are scheduled that leads to better scheduling decisions. For SI=50 and SI=60, the solutions of AILP are given by integration of ILP and AGS, as for some SIs, AGS are utilized when feasible solutions are not given by ILP, which weakens the profit enhancement and cost saving advantages of ILP.

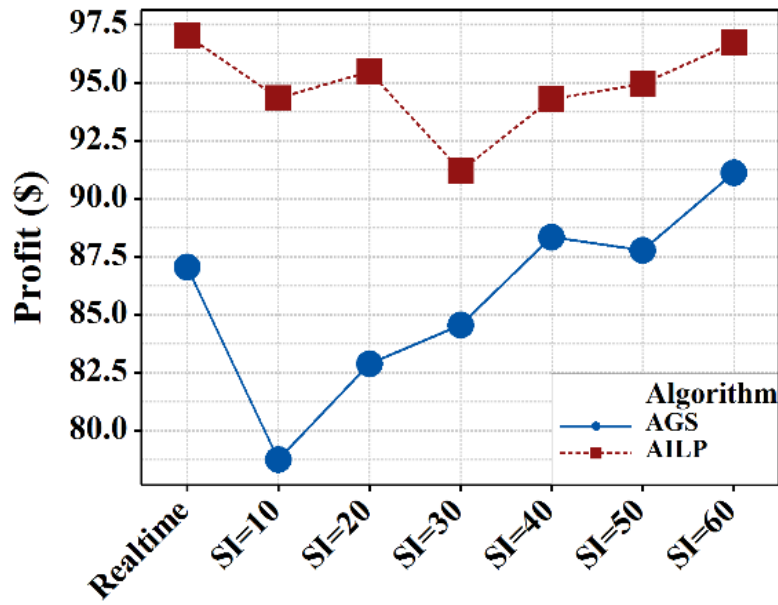


Figure 3.3: Profit of AILP and AGS.

As shown in Figure 3.4, for all scheduling scenarios, the median resource cost for AILP is \$135.3 while for AGS is \$145.4, which shows AILP gives 7.5% higher cost saving regarding median resource cost. The median profit made by AILP is \$95.0 while by AGS is \$87.0, which shows AILP creates 9.2% higher median profit than AGS. Results also show that the mean resource cost generated by AILP is \$135.3, which is 6.7% less than AGS while the mean profit created by AILP is \$94.9, which is 10.6% higher than AGS.

All of the obtained results show that AILP generates less resource cost and creates higher profit comparing to AGS, which proves the efficiency of AILP in achieving the cost saving and profit enhancement objectives. For real time scheduling, the benefit is

that more queries can be accepted for higher profit, while the drawback is that frequent scheduling consumes more computing resources. Thus, real time scheduling is not applicable when load of queries is high. For periodic scheduling, the more queries are collected, the better scheduling decisions can be made to save more resource cost and create higher profit. However, the drawback is that the longer the SI is, the more queries are rejected due to deadline constraints. This causes customer dissatisfaction due to higher request rejection rate, which also leads to reduction of market share. Therefore, long SI is not recommended even more resource cost can be saved. In order to achieve higher cost saving and profit enhancement, while at the same time accept more queries to serve more users for larger market share and higher user satisfaction, periodic scheduling with SI=20 is the best solution to execute the query workload.

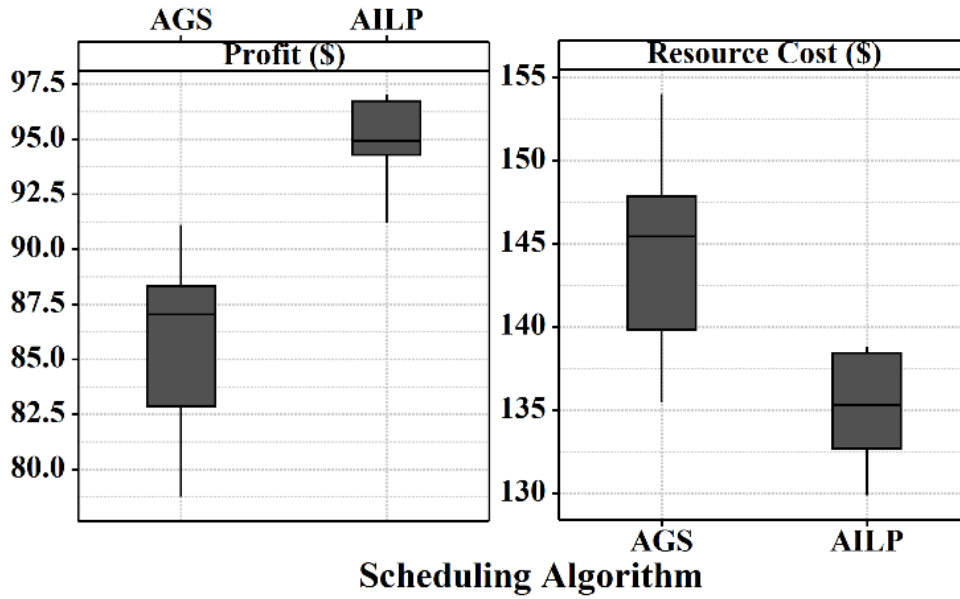


Figure 3.4: Profit and Resource Cost of AILP and AGS.

We further study the performance of AILP and AGS in profit enhancement and cost saving for different BDAAs using SI=20. Results are shown in Figure 3.5. We notice that resource cost and profit vary for different BDAAs, which is caused by variation of the number of accepted queries and their required resources. For different queries, query execution time can vary from minutes to hours. Resource cost generated by AILP is 1.9%, 2.4%, 15.5%, and 3.3% less than AGS for BDAA₁, BDAA₂, BDAA₃, and BDAA₄. Accordingly, the profit created by AILP is 3.5%, 4.3%, 26.2%, and 4.8% higher

than AGS. All the obtained results show AILP can generate less resource cost and create higher profit for each BDAA that further prove that AILP can better serve the cost saving and profit enhancement objectives.

Study of algorithm performance based on C/P metric

We use the C/P metric [74] to evaluate the performance of AILP and AGS. C/P is the quotient of resource cost and workload running time. Smaller C/P value indicates better performance of the scheduling algorithm. We can see that AILP has smaller C/P values for all scheduling scenarios, which indicates that AILP performs better than AGS, as shown in Figure 3.6.

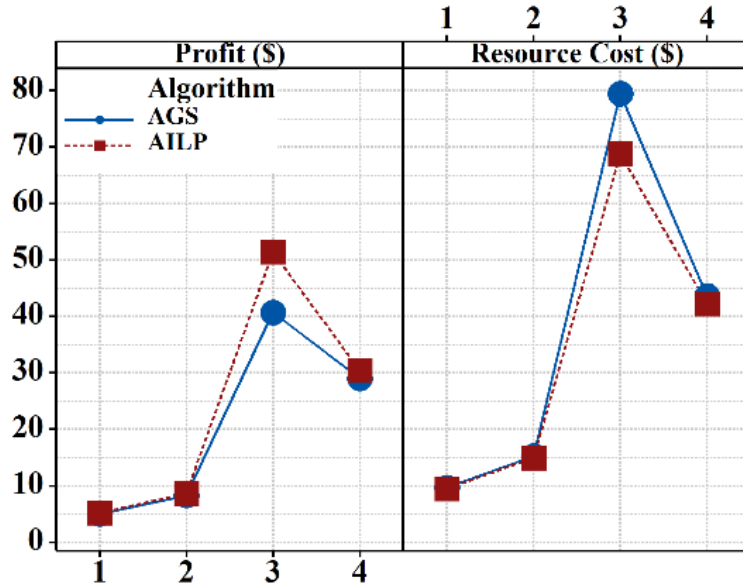


Figure 3.5: Profit and Resource Cost of BDAA for SI=20.

We use periodic scheduling with SI=20 as an example. For SI=20, results show that to execute the query workload, AILP uses 156.5 hours that is 78.9% longer than AGS and generates \$135.3 resource cost that is 9.3% less than AGS. This leads to a smaller C/P value of 0.9 for AILP, comparing to larger C/P value of 1.7 for AGS. For AGS, we notice that the trend of C/P keeps decreasing while SI increases, which indicates that AGS performs better when more queries are scheduled in each SI. For AILP, the trend of C/P fluctuates as the scheduling solution returned by AILP is an integration of ILP and AGS.

Study of ART

We record the ART for AILP and AGS to execute the query workload, as shown in Figure 3.7. For real time scheduling, a query is scheduled based on its arrival. There is

one query that is scheduled each time by AGS and AILP. Thus, ART is not a limitation factor. For periodic scheduling, variation of SI leads to variation of the number of queries to be scheduled in each SI. To avoid deadline violations when the number of queries is large, a scheduling timeout is set to limit the ART. The maximum value of the timeout is limited to 90% of SI to ensure that sufficient time is left for AGS to give scheduling solution when no feasible solution is given by ILP.

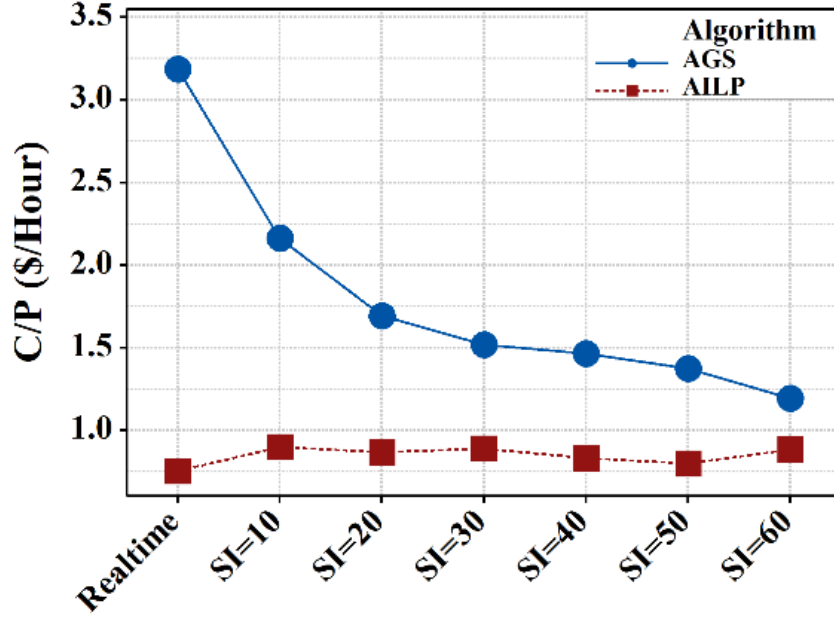


Figure 3.6: C/P of AILP and AGS.

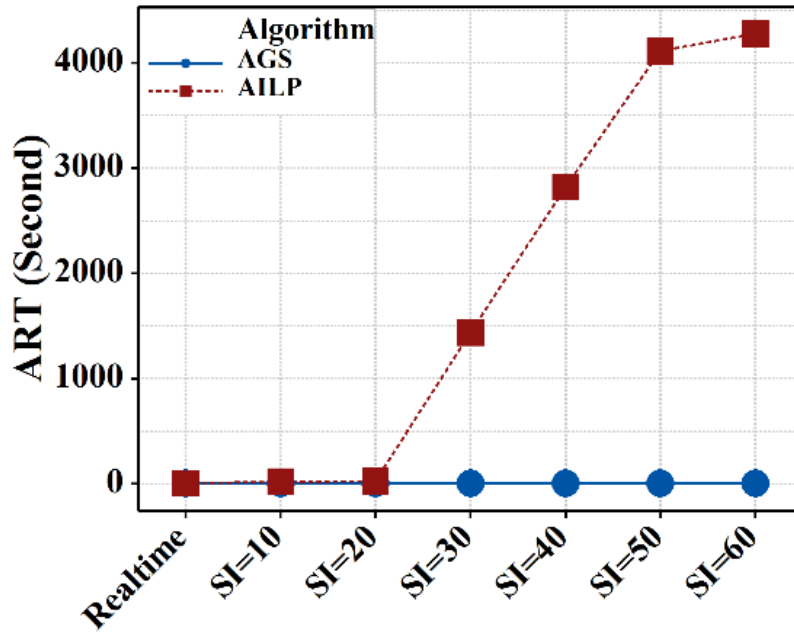


Figure 3.7: ART of AILP and AGS.

We made efforts to reduce the ART of ILP by utilizing two greedy algorithms to decide the input of VMs to Phase 1 and Phase 2 of ILP scheduling algorithm, which ensures that the given VMs have sufficient capacity to execute queries; meanwhile, the given number of VMs is close to the number of VMs of the optimal solution. In this way, ILP does not need to search large solution space, which greatly reduces the ART of ILP. For different query workload, SI can be adjusted to a suitable value based on the arrival rate of queries to allow AILP algorithm generating scheduling solutions to better achieve cost saving and profit enhancement objectives.

Results show that ART_{AILP} is longer than ART_{AGS} for all scheduling scenarios. For periodical scheduling, AGS takes milliseconds to give a scheduling decision for each SI and ART_{AGS} increases when SI increases. However, the increasing rate of ART_{AGS} is not as rapid as ART_{AILP} . For AILP, the increasing rate of ART_{AILP} decreases while SI increases as the ART cannot exceed the scheduling timeout, which is a constant value. Obtained results show that ART is not the limiting factor for AILP to serve as the scheduling algorithm of the AaaS platform as it can give scheduling solution within each SI. However, it is not recommended to use long SI. As the longer the SI is, the more requests with tight deadline are rejected by the admission controller, which is not consistent with the profit maximization objective of the AaaS platform. Moreover, it also decreases the user satisfaction and reduces the market share of AaaS providers.

3.5 Related Work

Our research focuses on provisioning effective and efficient algorithms to support the AaaS platform to deliver on-demand AaaS for various domains of users with SLA guarantees and at lower cost in order to enhance profit, enlarge market share, and improve user satisfaction. We compare our work with the most related works with respect to various parameters, as summarized in Table 3.5.

Sun et al. [12] propose a general-purpose analytic framework to provision cost-effective analytic solution with multi-tenancy support. They propose an SLA customization mechanism to satisfy diverse QoS requirements of tenants. However, their work does not consider admission control; thus, SLAs are at risk of violations. Moreover, they do not address resource scheduling algorithms with SLA guarantees on deadline and budget and also do not address profit enhancement purpose of AaaS providers.

Mian et al. [74] focus on resource provisioning for data analytic workloads in a public cloud that aims to determine the most cost-effective resource configuration using greedy search heuristic. Their work schedules queries for one application, which does not serve general data analytic purpose. Moreover, the SLA they considered is average query response time and SLA violation is allowed for a cheaper resource cost, which is different from our SLA guaranteeing perspective. We believe that SLA violations can significantly decrease user satisfaction and reputation of AaaS providers. Therefore, SLA violations should be avoided.

Table 3.5: Related Work

Parameter	Related Work						Our Work
	[12]	[74]	[119]	[21]	[93]	[94]	
<i>AaaS</i>	Y	N	N	Y	N	N	Y
<i>Cost saving</i>	Y	Y	Y	Y	Y	Y	Y
<i>Profit enlarging</i>	N	N	N	N	N	N	Y
<i>SLA guarantee</i>	Y	N	Y	Y	Y	N	Y
<i>Admission control</i>	N	N	Y	N	N	N	Y
<i>Budget</i>	N	N	N	N	Y	Y	Y
<i>Deadline</i>	N	N	N	Y	Y	Y	Y
<i>Scalable and elastic resource provision</i>	N	Y	Y	Y	N	Y	Y

Y: covered; N: not covered

Garg et al. [119] manage resources for heterogeneous workload in a datacenter. They focus on resource scheduling problem for mixed workloads instead of data analytic workloads. They propose an admission control and resource scheduling algorithm, which considers deadline-constrained scheduling while we consider budget-constrained scheduling as well, which is the market feature of delivering analytics as a service in cloud computing environments. Zulkernine et al. [21] propose the conceptual architecture of cloud-based analytics as a service, which does not consider admission control. Their work focuses on data analytics for workflow applications.

Scheduling and resource provisioning for MapReduce tasks is a well-explored area, which targets at a specific application model and does not serve general data analytic

purpose. Alrokayan et al. [93] propose cost-aware and SLA-based algorithms to provision cloud resources and schedule MapReduce tasks under budget and deadline constraints. They adopt Lambda architecture and focus their current work on the batch layer. Mattess et al. [94] propose an algorithm to dynamically provision resources to meet soft deadline of MapReduce tasks while minimizing the budget. Both of the above works do not consider admission control for SLA guaranteeing purpose.

There are some works focus on delivering specific analytics as a service and provisioning query processing techniques, which give support to the AaaS platform and enrich the BDAAAs that the AaaS platform can provide to users. Chen et al. [32] address the challenges in delivering continuous analytics as a service to analyze real-time events. Barga et al. [33] propose Daytona to offer scalable computation for large scale data analytics, which focuses on spreadsheet applications. Agarwal et al. [58] propose BlinkDB, which enables approximate query processing on data samples of large datasets with bounded response time and bounded errors.

3.6 Conclusions and Future Work

Data analytics has significant benefits in decision making and problem solving for various domains. To support the AaaS platform that offers data analytics as a service to various domains of users as consumable services, our research focuses on proposing effective and efficient admission control and resource scheduling algorithms. The aim is to allow AaaS providers delivering AaaS with SLA guarantees to increase market share, improve reputation, and enhance profit. We have proposed the architecture and models of the AaaS platform and conducted experiments to evaluate the performance of the admission control and resource scheduling algorithm. Experiments are conducted based on different scheduling scenarios, which are non-periodic (real time) scheduling and periodic scheduling with varied scheduling intervals. Results have shown that our admission control algorithm successfully admits queries based on QoS requirements to allow scheduling algorithms giving SLA guarantees. Moreover, AILP, which is an integration of AGS and ILP, can save more resource cost and create higher profit while overcoming the limitation of ILP in ART. As part of the future work, we will (1) continue working on proposing efficient admission control and resource scheduling algorithms; (2) study the effect of application profiling in the performance of algorithms;

and (3) study data sampling techniques that allow query processing on sampled datasets for quicker response time and higher cost saving.

Acknowledgments

We thank Mohsen Amini Salehi, Yun Yang, Satish Srirama, and all the members at the CLOUDS Lab for their valuable comments and suggestions to improve the work.

4 SLA-Based Profit Optimization of Resource Management

In this chapter, we focus on Problem Scenario 2 and optimizing resource management solutions. This chapter was previously published¹ which was extended from paper². The major contributions of this chapter are optimal resource management algorithms that maximize profits through effective query admission, dynamic resource provisioning, query time minimization of resource configurations, and SLA guarantees on budgets and deadlines. The proposed algorithms enable queries to trade-off accuracy of query results for faster response times and reduced resource costs under given budget or deadline constraints.

Specifically, Chapter 4 provides optimal resource management algorithms that enhance the solutions described in Chapter 3 since they can be sub-optimal due to the two-phase resource scheduling. We refine the modeling and formulation of the profit optimization problem and enhance the admission control and resource scheduling to support data splitting-based methods that allow to process smaller split datasets as representative data to tackle big data analytics challenges under budget and deadline constraints. Experimental results show that the proposed algorithms significantly increase profits, reduce resource costs, increase query admission rates, and decrease query processing times compared to non-data splitting-based resource scheduling algorithms.

This chapter is originally derived from the following publications:

¹Y. Zhao, R. N. Calheiros, G. Gange, J. Bailey, and R. O. Sinnott, “SLA-based profit optimization for resource scheduling of big data analytics-as-a-service in cloud computing environments,” *IEEE Transactions on Cloud Computing* (TCC), 2018.

²Y. Zhao, R. N. Calheiros, J. Bailey, and R. O. Sinnott, “SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments,” in *Proceedings of IEEE International Conference on Big Data*, pp. 432-441, 2016.

SLA-Based Profit Optimization Resource Scheduling for Big Data Analytics-as-a-Service Platforms in Cloud Computing Environments

Yali Zhao¹, Rodrigo N. Calheiros², Graeme Gange³, James Bailey¹,
Richard O. Sinnott¹

¹The University of Melbourne, Australia

yalizhao.alice@gmail.com, {baileyj@, rsinnott@}unimelb.edu.au

²Western Sydney University, Australia

r.calheiros@westernsydney.edu.au

³Monash University, Australia.

Graeme.Gange@monash.edu

Abstract

The value that can be extracted from big data greatly motivates users to explore data analytics technologies for better decision making and problem solving in various application domains. Data analytics solutions can be expensive due to the demand for large-scale and high-performance computing resources. To provision online big data Analytics-as-a-Service (AaaS) to users in various domains, a general purpose AaaS platform is required to deliver on-demand services at low cost and in an easy to use manner. Our research focuses on proposing efficient and automatic admission control and resource scheduling algorithms for AaaS platforms in cloud environments. In this paper, we propose scalable and automatic admission control and resource scheduling algorithms, which maximize profits and minimize query times, effectively admit data analytics requests, and dynamically provision resources, while satisfying QoS requirements of queries with Service Level Agreement (SLA) guarantees. Moreover, the proposed algorithms enable queries to trade-off accuracies for faster response times and less resource costs by adopting a data splitting based method to process smaller split datasets as representatives of the original big datasets. We conduct extensive experiments to evaluate the proposed admission control and profit optimization scheduling algorithms. Experimental evaluation shows the algorithms perform significantly better compared to the state-of-the-art algorithms in enhancing profits,

reducing resource costs, increasing query admission rates, and decreasing query response times.

Keywords: Admission Control; Resource Scheduling; Service Level Agreement; Analytics-as-a-Service; Data Splitting; Big Data; Cloud Computing

4.1 Introduction

The value that can be extracted from big data greatly motivates users to explore data analytics techniques and solutions [1, 2]. Big data analytics is an emerging trend for decision-making and problem solving in many application domains, such as healthcare and medication. Cloud computing [129, 130] greatly eases and benefits big data analytics by elastically provisioning heterogeneous and on-demand resources and services at infrastructure, platform, and application levels, which are provisioned in a pay-per-use basis with Service Level Agreement (SLA) guarantees. Big Data Analytics Applications (BDAAAs) are usually targeted to specific domains and typically used by large enterprises [12]. BDAAAs require large-scale computing, which can incur significant financial costs based on the consumed resources. As such, BDAAAs are usually used by large enterprises or major research projects and are not suitable for occasional or individual usage.

To facilitate big data analytics for various domains, a general purpose Analytics-as-a-Service (AaaS) platform is required to lead the market and deliver on-demand services to users at low cost and in an easy to use way [6]. Our proposed AaaS platform brokers BDAAAs from third party application providers and commodity resources, such as Virtual Machines (VMs) and storage from cloud resource providers. The AaaS platform automatically manages cloud resources for BDAAAs based on user demands and provisions AaaS as consumable services. We focus especially on BDAAAs that typically process read-only queries and leave out data consistency and security issues. The AaaS delivery model greatly benefits users with limited expertise to set up and manage large-scale cloud computing environments and enjoy on-demand AaaS with guaranteed SLAs on a pay per use basis.

Resource scheduling is a core function for AaaS platforms that aims to maximize profits by optimizing resource and query scheduling, increase market share by admitting more queries, and improve user satisfaction by delivering on-demand AaaS

with SLA guarantees. AaaS platforms schedule resources to execute queries with guaranteed SLAs on QoS requirements of budgets and deadlines, which allows users to control analytics costs and get timely results for efficient decision-making. However, resource scheduling for big data analytics on AaaS platforms must tackle several challenges:

- Data analytics for big data with large volumes (i.e., terabytes, petabytes) can take considerable time. This requires efficient resource management and task scheduling techniques that are capable to return query results before user-specified deadlines for fast and efficient decision-making, especially for application domains requiring timely decisions.
- Big data analytics requires large-scale compute and storage resources that can incur considerable resource costs. It is a necessity for users to obtain query results under controllable costs, especially for users with limited budgets.
- Resource requirements of online analytics requests fluctuate over time, which raises the requirement for automatic resource provisioning to react to dynamic query demands, which can be stochastic.
- Guaranteeing SLAs for AaaS is important for user satisfaction. Violating SLAs not only incurs penalty costs but also decreases user satisfaction. Effective and efficient admission control and resource scheduling are required to minimize/avoid SLA violations.
- Query response time is a major performance indicator of service delivery and should be minimized for better user experience. Transferring big data for processing can take considerable time and incur high resource costs and hence should be minimized. Data locality-aware scheduling techniques should be utilized to move the computing resources to the data by creating cloud resources at datacenters where the datasets reside.
- Maximizing profits and minimizing query response times while guaranteeing SLAs of service delivery should be supported. This is a multi-objective optimization problem. The profit optimization problem requires efficient and accurate modeling and formulation for optimal resource scheduling.
- Data analytics requests with tight budgets and deadlines cannot be admitted due to SLA violations. Lack of support to process such requests can cause significant

reduction of profit, decrease user satisfaction, and lead to loss of customers. Among existing data analytics techniques is data splitting [57] that enables users to trade-off accuracy for faster response times and less resource costs by processing smaller data samples as representatives of the original datasets. Efficient adoption of such data processing techniques in resource scheduling is required to process big data with limited budgets and times.

To tackle the above challenges, our research focuses on proposing efficient and automatic admission control and resource scheduling solutions to support AaaS platforms, to not only maximize the profit for AaaS providers, but also to minimize query processing times while guaranteeing SLAs for AaaS users. The **major contributions** of our work are:

- We model and formulate a multi-objective profit optimization and query time minimization resource scheduling problem.
- We propose admission control and profit optimization scheduling algorithms that are able to admit queries whose QoS requirements can be satisfied to deliver services with SLAs guarantees; maximize the profits for AaaS platforms by optimizing resource utilization; and minimize query response times with time-efficient query execution solutions.
- The optimization algorithms utilize automatic scheduling mechanisms that allow the AaaS platform to provide online data analytics services that react to dynamic variation of resource requirements based on realtime queries.
- The optimization scheduling algorithms utilize scalable scheduling mechanisms, which elastically lease under-provisioned cloud resources while releasing over-provisioned resources to optimize utilization and minimize cost.
- The scheduling algorithms adopt data-aware scheduling to ensure the best data processing locality to minimize the network transfer times and costs.
- The scheduling algorithms enable users to trade-off accuracies for reduced response times and less costs. Applying data splitting and sampling techniques provides significant performance improvement for resource scheduling that enables AaaS platforms to process data samples compared to processing the original big datasets.
- The scheduling algorithms support flexible query dependencies. A query can vary from an independent task to a complex dataflow. Queries that rely on results of

other tasks can only be executed after their dependent tasks complete. Such flexible support for task dependencies enables the AaaS platform to handle tasks of different complexities.

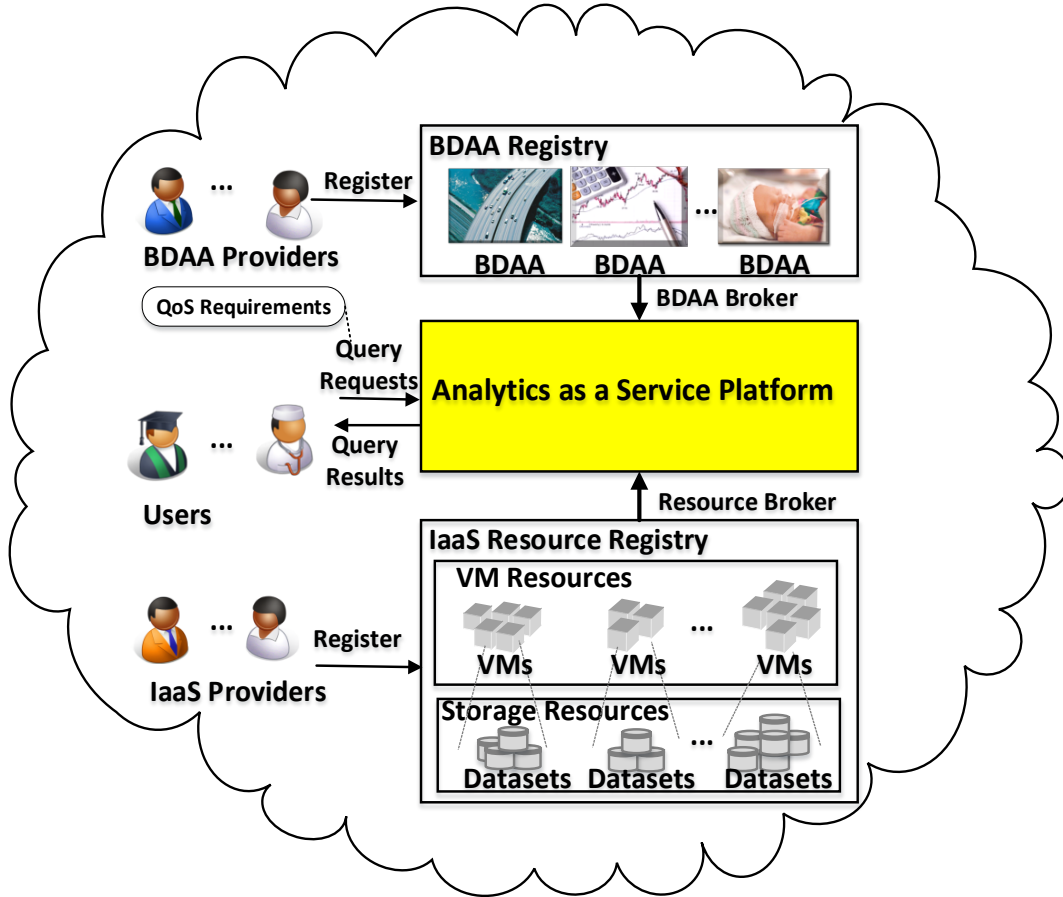


Figure 4.1: Overview of the AaaS Environment.

We conduct extensive experiments to evaluate the admission control and resource scheduling algorithms. The results show that the admission control algorithms effectively admit queries for SLA guarantees and the optimization scheduling algorithms perform significantly better with regards to saving costs, enhancing profits, and reducing query processing times in comparison to related state-of-the-art algorithms. Results also show that after applying data splitting-based scheduling mechanisms, there is a significant performance improvement including enhancing profits, reducing resource costs, increasing admission rates, and reducing query processing times over traditional resource scheduling mechanisms without data splitting.

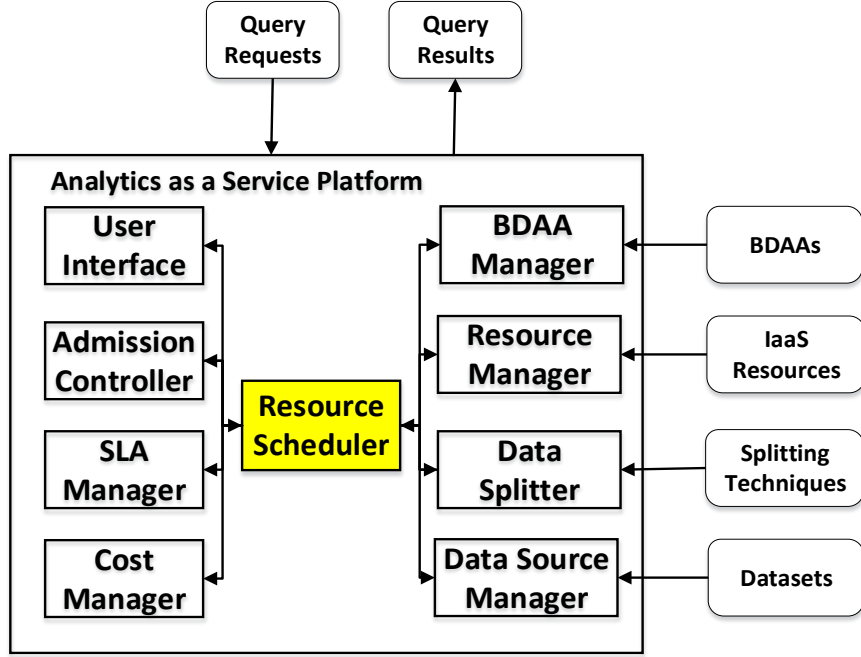


Figure 4.2: Architecture of the AaaS Platform.

This paper is a significant extension of our previous work [11]. Compared to our earlier work, we refine and enhance the modeling and formulation of the multi-objective optimization problem for resource scheduling to accommodate new features of task dependency and data splitting based scheduling. Moreover, we refine the architecture and framework of the AaaS platform by adding data splitter to allow splitting of large datasets and processing smaller data samples for quicker response times and less costs. We also redesign the data splitting-based admission control and profit optimization algorithms to allow data splitting and task dependency-based admission control and resource scheduling. Furthermore, we redesign the experiments and conduct extensive experiments to evaluate the performance of the data splitting-based scheduling algorithm and non data splitting-based methods. We survey the state-of-the-art research works and compare our work to these related works with regards to various performance parameters. In summary, this paper makes a significant contribution on the applicability, efficiency, and performance improvement of the data splitting-based admission control and profit optimization scheduling algorithms for AaaS platforms in cloud computing environments.

4.2 Problem Statement

4.2.1 Overview of the AaaS Environment

An overview description of the AaaS Environment is shown in Figure 4.1.

Users submit queries online to the AaaS platform requesting particular data analytics services supported by specified BDAAAs. These queries contain query specifications including QoS requirements on budgets and deadlines and data locations where to-be-analyzed datasets reside. The AaaS platform brokers cloud resources from third party Infrastructure-as-a-Service (IaaS) cloud providers while it also brokers BDAAAs from third party application providers to provide analytics services based on application requirements of data analytics requests. BDAAAs are maintained in a BDAA registry, which is a catalog for BDAA providers to register for BDAAAs to the AaaS market and also a directory for AaaS platforms to search for required BDAAAs. At any time, a BDAA can be registered and deregistered from the BDAA registry. Thus, the AaaS platform needs to search for availability of the requested BDAAAs before admitting queries. Similarly, cloud resources are registered and maintained in an IaaS resource registry. After the AaaS platform admits queries based on its admission policy, it makes profit optimization resource and query scheduling decisions to execute queries. Admitted queries are executed on allocated cloud resources by adopting query scheduling and resource provisioning solutions. Finally, the AaaS platform provisions AaaS to users by automatic and dynamic resource provisioning and query scheduling and deploys BDAAAs on existing cloud resources for subsequent query processing.

4.2.2 Architecture of the AaaS Platform

The architecture for the AaaS platform is shown in Figure 4.2, which includes the following components.

User Interface serves as a mechanism for users and the AaaS platform to interact with each other. Users submit query specifications and negotiate SLAs with the AaaS platform. Details of the specifications are introduced in the modeling section. The AaaS platform outputs admission decisions and query results via the user interface.

Admission Controller is responsible for query admission. Queries that can be admitted must satisfy the following conditions: 1) the BDAAAs requested by queries are available; 2) QoS requirements of queries can be satisfied based on current resource

capacity and configuration; 3) to-be-processed datasets are accessible from data storage.

SLA Manager gets the queries after the admission control and builds SLAs for the admitted queries to ensure the AaaS platform can guarantee SLAs during service delivery to avoid SLA violations that incur additional penalty costs and cause user dissatisfaction.

Cost Manager manages all the costs in the AaaS platform including BDAA costs; cloud resource costs, AaaS service costs, and penalty costs. It provides pricing information to allow the Admission Controller to make performance estimation for query admission.

Resource Manager brokers IaaS resources from third party IaaS resource providers and keeps a catalog of all available resources from the different IaaS providers. The Resource Manager keeps monitoring the status of VMs, which are created, terminated, in-use, and idle. This information allows the Query Scheduler to take different actions accordingly, e.g. sending queries to VM instances that are idle for load balancing, terminating idle VM instances to save costs.

BDAA Manager is responsible for brokering BDAA resources from third party service providers based on user requests and maintaining application information. Information of BDAAs is maintained and updated in the BDAA Registry for application searching and selection.

Data Source Manager is responsible for managing to-be-processed datasets factoring in their size, type, and location. We utilize cloud resources in datacenters where datasets reside to reduce data transfer times and costs.

Data Splitter is responsible for splitting large datasets to obtain smaller split datasets for data analytics when budgets and deadlines are limiting factors. The data splitting can be achieved by different approaches such as data splitting and sampling. For queries that do not require full accuracy based on processing the entire datasets, the accuracy can be traded-off for faster response times and less resource costs by processing a less complete and hence less accurate subset of the data. For queries that cannot be admitted due to tight deadlines or limited budgets, if queries allow approximate query processing, the AaaS platform checks whether the query can be admitted by estimating the splitting time and cost together with the data processing time and cost compared to user-specified query deadlines and budgets. If applying data splitting techniques can process

the data with SLA guarantees on QoS requirements, such queries can be admitted for processing.

Query Scheduler is the central component of the AaaS platform and coordinates the other components. It makes query scheduling and resource provisioning decisions based on realtime resource and BDAA information in the AaaS platform. It aims to make optimization scheduling decisions to maximize profits, increase query admission rates, guarantee SLAs, while reacting to the dynamism of query workloads and resource capacities. The query scheduler enables data locality-aware and dependency-based scheduling while supporting data splitting-based scheduling for faster response times and less resource costs.

4.2.3 Problem Modelling

The profit optimization resource scheduling problem aims to find the optimal resource scheduling solutions to maximize profits for the AaaS platform and minimize query times, while delivering satisfactory services to users with SLA guarantees.

We consider a set of M queries $Qs = \{Q_1, Q_2, \dots, Q_M\}$ submitted to the AaaS platform. A query can be a scan query, i.e., select the average body fat from a patient table, or a complex query, i.e., a machine learning algorithm to analyze the factors leading to lung cancer, etc. These queries request for a set of P Big Data Analytics Applications $BDAAs = \{BDAA_1, BDAA_2, \dots, BDAA_P\}$ that execute on a set of N resources $Rs = \{R_1, R_2, \dots, R_N\}$. A BDAA can be a healthcare application running on top of data analytics framework Spark [131] to analyze patient health data. A resource can be a VM, storage, or a cluster of VMs, etc. Different cost policies are applied in the AaaS platform, such as policies for query costs and penalty costs.

Query $Q = \{SLA, CR, BDAA, CH, DE, AC\}$, where SLA denotes QoS requirements of budget (the maximum cost to execute a request) and deadline (the latest time to return the analytics result); CR denotes the requested cloud resources to execute a query task, i.e., individual VM or VM cluster; $BDAA$ indicates the specific data analytics applications that users request to analyze the data, i.e., healthcare application; CH specifies the characteristics of the dataset, such as data size, data type, data distribution, and data locality; DE denotes the dependency requirement, i.e., father tasks and child tasks; AC denotes whether the accuracy of the query can be sacrificed, i.e., 95% accuracy. If users allow sacrificing accuracy during data analytics procedures when

budget or deadline of a query is a limiting factor, the AaaS platform may enable processing representative datasets using data splitting techniques. To avoid expensive data transfer costs and long data transfer times, we move the computing resources to the data, hence we process data by creating VMs in the cloud datacenters where the big datasets are stored. We assume the data is pre-stored in datacenters that storage and data transfer costs are paid directly by users.

Big Data Analytics Application $BDAA = \{AT, P\}$, where AT represents the application type while P represents the application profile, which contains mappings of the BDAA costs, required resource times including query times, sampling times, and splitting times, and resource configurations for different query requests. Obtaining profiles of heterogeneous BDAA applications requires expert knowledge from various application domains. Thus, we assume reliable BDAA profiles are provided by third-party BDAA providers. BDAAs provisioned by third party application providers should be accessible through a uniform interface to ease BDAA brokering and utilization.

Resource $R = \{RT, REC, NR, CC\}$, where RT denotes the resource type (e.g., memory optimized Amazon EC2 VM instances, such as an r3.large instance or a cluster of r3.large instances), REC represents the resource capacity (e.g., CPU, memory, storage, etc.) provided by the resource, NR represents the number of instances required to execute a query, while CC is the cost of each resource.

Cost $C = \{RC, BC, PC, ARC, PRO\}$, where RC denotes the overall resource cost of the AaaS platform, BC denotes the cost of BDAA applications that is assumed as a constant cost charged by BDAA providers, PC represents the penalty cost if any SLA of a given query is violated, ARC denotes the overall cost that users pay the AaaS platform for utilization of the data analytics services, and PRO denotes the overall profit made by the AaaS platform. We assume a constant service fee is charged for each query request depending on specific requested cloud resources, BDAA, and QoS requirements. Hence, ARC is a constant value.

4.2.4 Problem Formulation

The profit optimization scheduling problem is an NP-complete problem that can be polynomially reduced to an Integer Linear Programming (ILP) problem. We formulate the profit maximization resource scheduling problem based on the ILP model [120, 118]. Based on the cost model as introduced in the previous section, we can know ARC

is a constant value. We aim to provide scheduling solutions with SLA guarantees on QoS requirements. Thus, PC is zero if SLAs are guaranteed. The PRO is calculated as $PRO = ARC - RC - AC - PC$. As ARC and AC are constant while PC is zero while SLAs are guaranteed, the objective of maximizing PRO is to minimize RC while minimizing the query processing times and delivering AaaS with SLA guarantees.

To achieve the scheduling objective, we have two individual objectives to fulfill: Objective A and Objective B . $A = \text{minimize}(\sum_{j=1}^n C_j * t_j)$, where j represents a cloud resource such as a VM or a cluster of VMs; n represents a set of resources; t_j is the number of hours to be used for a resource j ; while C_j is the hourly cost of the resource j . Objective A aims to minimize the resource cost RC , calculated as the total costs of different resources, which is the product of purchased resource time t_j and the unit resource cost C_j . Objective $B = \text{minimize}(\sum_{i=1}^m s_i)$, where i represents a query request i ; m denotes a set of query requests, while s_i is the start time of the query i . Objective B aims to find a query scheduling solution with the cost minimized resource configuration that can execute queries at the earliest time. This enables future queries to start execution at the earliest time s_i to reduce t_j and hence allows cost saving and minimizes the query times for quicker responses to improve the performance of delivered services. The symbols of the formulation are defined in Table 4.1.

We formulate the multiple Objective Z as a combination of Objective A and Objective B , which aims at minimizing resource costs while choosing a scheduling solution that starts query execution at the earliest time, subject to constraints (1)-(25). The combination leads to a standard lexicographic optimization problem [121]. The importance of individual objectives contributing to the profit optimization resource scheduling problem is $A > B$. Coefficient F_0 is assigned to Objective A to guarantee the aggregated problem with minimization of all individual objective functions is consistent to the original profit optimization problem that any change of A dominates the changes of B [118, 121].

Objective Z :

$$Z = \text{minimize} (F_0 * \sum_{j \in n} (C_j * t_j) + \sum_{i \in m} s_i)$$

Subject to:

$$\sum_{i \in m, j \in n} (RRT_{ij} * x_{ij}) \leq t_j + T_j \quad (1)$$

$$x_{ij} = \begin{cases} 1, & i \text{ is scheduled on } j \\ 0, & \text{otherwise} \end{cases}, \forall i \in m; j \in n \quad (2)$$

$$b_{ik} + b_{ki} \leq 1, \forall i, k \in m \quad (3)$$

$$b_{ik} = \begin{cases} 1, & i \text{ executes before } k \\ 0, & \text{otherwise} \end{cases}, \forall i, k \in m \quad (4)$$

$$b_{ik} + b_{ki} - x_{ij} - x_{kj} \geq -1, \forall i, k \in m; j \in n \quad (5)$$

$$s_i - s_k + F_1 * b_{ik} \leq F_1 - RRT_{ij}, \forall i, k \in m; j \in n \quad (6)$$

$$s_i + F_2 * x_{ij} \leq F_2 + D_i - RRT_{ij}, \forall i \in m, j \in n \quad (7)$$

$$s_i - t_j + F_3 * x_{ij} \leq F_3 - RRT_{ij} + EPT_j, \forall i \in m, j \in n \quad (8)$$

$$s_i + x_{ij} * F_4 \geq F_4 + EAT_j, \forall i \in m, j \in n \quad (9)$$

$$C_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (10)$$

$$\sum_{i \in m, j \in n} x_{ij} = 1 \quad (11)$$

$$x_{ij} = 0, \text{ if } j \text{ cannot access dataset of } i, \forall i \in m, j \in n \quad (12)$$

$$b_{ki} = 1, \text{ if } i \text{ needs to process dataset split by } k, \forall i, k \in m \quad (13)$$

$$b_{ki} = 1, \text{ if } i \text{ is a dependent task of } k, \forall i, k \in m \quad (14)$$

$$\begin{cases} x_{ij} = 1 \\ x_{kj} = 1 \end{cases} \implies \begin{cases} b_{ik} = 1, b_{ki} = 0 \\ b_{ik} = 0, b_{ki} = 1 \end{cases}, \forall i, k \in m, j \in n \quad (15)$$

$$b_{ik} = 1 \Rightarrow s_i + RRT_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (16)$$

$$F_1 \geq \max (s_i + RRT_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (17)$$

$$s_i * x_{ij} + RRT_{ij} \leq D_i, \forall i \in m, j \in n \quad (18)$$

$$F_2 \geq \max (s_i + RRT_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (19)$$

$$x_{ij} = 1 \Rightarrow s_i + RRT_{ij} \leq t_j + EPT_j, \forall i \in m, j \in n \quad (20)$$

$$F_3 \geq \max (s_i - t_j + RRT_{ij} - EPT_j) + 1, \forall i \in m, j \in n \quad (21)$$

$$x_{ij} = 1 \Rightarrow s_i \geq EAT_j, \forall i \in m, j \in n \quad (22)$$

$$F_4 \leq \min (s_i - EAT_j) - 1, \forall i \in m, j \in n \quad (23)$$

$$T_j = \begin{cases} LRT_j, & j \text{ is created} \\ -RCT_j, & \text{otherwise} \end{cases}, \forall j \in n \quad (24)$$

$$EAT_j = \begin{cases} \begin{cases} TFT_j, & j \text{ is executing queries} \\ CT_j, & \text{otherwise} \end{cases}, & j \text{ is created} \\ CT_j + RCT_j, & \text{otherwise} \end{cases}, \forall j \in n \quad (25)$$

Resource Capacity Constraints

Resource capacity constraints are used to ensure that the sum of required resource times of queries for resource j should be less than or equal to the available resource

time of resource j , as shown in (1), where x_{ij} is a binary variable indicating whether query i is scheduled to resource j ; RRT_{ij} is the Required Resource Time (RRT) to execute query i on resource j ; t_j is the number of hours requested to purchase for resource j . When query i is scheduled to resource j , $x_{ij} = 1$; otherwise, $x_{ij} = 0$, as shown in (2). For created resources, T_j is the Left Resource Time (LRT) before the end of purchased time in the previous schedule, denoted as LRT_j , which can be used to execute queries in the current schedule; while for newly created resources, T_j is a negative value that is used to deduct the Resource Creation Time (RCT) from t_j , denoted as $-RCT_j$, as shown in (24).

Table 4.1: Definition of Symbols

Symbol	Definition
i	A specific query
j	A specific resource
m	The set of queries
n	The set of resources
t_j	The time to be purchased for resource j
T_j	The available resource time of resource j .
LRT_j	The remaining time of resource j in previous schedule
RRT_{ij}	The required resource time for query i on resource j
x_{ij}/x_{kj}	It indicates whether query i / query k is assigned to resource j
C_j	The unit cost of resource j
s_i/s_k	The start time of query i / query k
b_{ik}/b_{ki}	It indicates whether query i executes before query k
D_i	The deadline of query i
B_i	The budget of query i
C_{ij}	The cost of executing query i on resource j
EAT_j	The earliest time that resource j is available to execute queries
TFT_j	The query finish time on created resource j
CT_j	The clock time of created resource j
RCT_j	The creation time of resource j

Query Execution Sequence Constraints

Query execution sequence constraints define the execution sequence to ensure the unique execution sequence of queries for the optimization solution, as shown in (3), where b_{ik} is a binary variable, which indicates whether query i is scheduled before query k . If query i is scheduled before query k , $b_{ik} = 1$; otherwise, $b_{ik} = 0$.

Query Deadline Constraints

Deadline constraints are used to ensure analytics solutions are returned before deadlines, as shown in Constraints (4) - (9). Constraint (4) is used to ensure the unique execution order of query i and query k by allowing only one of b_{ik} and b_{ki} to be 1. Constraint (5) ensures that query i must be executed either before query k or after query k when they are executed on the same resource j , which is derived from non-linear constraint (15) through linearization. Constraint (15) ensures that if query i and query k are scheduled on resource j , either $b_{ik} = 1, b_{ki} = 0$ (query i executes before query k) or $b_{ik} = 0, b_{ki} = 1$ (query i executes after query k). Constraint (6) is derived from non-linear constraint (16) using Big M method, which is used to ensure if $b_{ik} = 1$, query i should finish execution before query k starts execution, where F_1 is a sufficient large constant satisfying (17) [6]. Constraint (7) is used to ensure that if query i starts execution on resource j at s_i , it should finish before its deadline D_i , derived from non-linear constraint (18), where F_2 is a sufficient large constant satisfying (19). Constraint (8) is derived from non-linear relationship (20), where F_3 is a sufficient large constant satisfying (21), which ensures that if query i is scheduled on resource j , its finish time has to be earlier than or equal to the end of the purchased time of resource j , which is the sum of previous purchased end time EPT_j of resource j and the new purchased time t_j . Constraint (9) is derived from non-linear relationship (22), where F_4 is a sufficient small constant satisfying (23), which is used to ensure that if query i is assigned to resource j , it should start execution after EAT_j that is the Earliest Available Time (EAT) of resource j . If resource j is created while it has queries executing on it, EAT_j is the Task Finish Time (TFT) of resource j , denoted as TFT_j ; otherwise, EAT_j is the clock time of resource j , denoted as CT_j . If resource j is not created, EAT_j is the sum of CT_j and RCT_j , as shown in (25).

Query Budget Constraints

Budget constraints are used to ensure that the execution cost of query i on resource j does not exceed the budget B_i , as shown in (10), where C_{ij} is the cost of executing query i

on resource j .

Query Scheduling Times Constraints

As shown in (11), the constraints on query scheduling times ensure that the sum of x_{ij} for a query must be 1 to ensure that the query i is scheduled to one resource for execution to avoid SLA violations.

Data Locality Constraints

Data locality constraints, as shown in (12) are used to ensure that queries can only be submitted to resources where to-be-processed datasets are accessible to avoid expensive data transfer costs and long data transfer times. If resource j does not have access to the dataset of a query i , $x_{ij} = 0$. This indicates that query i cannot be scheduled to resource j . For resources that have access to the dataset of query i , x_{ij} will be allowed to be 1 only if such assignment can lead to a profit optimization scheduling solution.

Data Splitting Constraints

Data splitting on very large datasets is required when data analytics tasks are restricted with tight deadlines or limited budgets. Data splitting and sampling techniques aim to generate small data representatives of original large datasets. Processing smaller data samples can reduce resource costs while returning analytics results in a time-efficient manner. If large datasets are to be processed in such way, $b_{ki} = 1$, as shown in constraint (13), which indicates a data processing task i has to be executed after data splitting task k finishes the data splitting job.

Query Dependency Constraints

Query dependency constraints are used to define the task dependencies, as shown in (14), $b_{ik} = 1$ is set when query i depends on the execution results of query k . This is to ensure a query can only start execution when its dependent queries finish execution. Query dependency constraints allow defining simple task dependencies, such as bag of tasks, as well as complex task dependencies, such as dataflows.

4.3 Admission Control and Resource Scheduling

In this section, we introduce the admission control and resource scheduling algorithms, which admit queries and make resource scheduling decisions to allow the AaaS platform to achieve profit optimization objectives with minimized query processing times and SLA guarantees.

4.3.1 Admission Control

Admission control is the prerequisite for the AaaS platform to make effective and efficient query and resource scheduling decisions. The admission controller aims to maximize the query admission rate for higher market share while ensuring all the admitted queries can be successfully executed with SLA guarantees. It only admits queries whose QoS requirements can be satisfied with provisioned BDAAAs and resources.

Algorithm 1: Admission Control Algorithm

Input: queries submitted by users

Output: admission or rejection decisions of queries

```
1: for each query  $\in$  submittedQueries
2:   requestedBDAA  $\leftarrow$  search BDAA registry
3:   if requested BDAA is available && dataset is accessible
4:     resConfigs  $\leftarrow$  search resource registry
5:     if resConfigs > 0
6:       for each configuration  $\in$  resConfigs
7:         EET = calculateEET
8:         EC = calculateEC
9:         if EET < deadline && EC < budget
10:          admQueries  $\leftarrow$  admit query
11:        end if
12:      else if query allows sacrifice accuracy
13:        apprQueries  $\leftarrow$  get approximate queries
14:        for each query in apprQueries
15:          DSC  $\leftarrow$  estimate resource cost for splitting
16:          DST  $\leftarrow$  estimate time for data splitting
17:          DPC  $\leftarrow$  estimate resource cost for splitting
18:          DPT  $\leftarrow$  estimate the splitting time
19:          if DSC+DPC < budget && DST+DPT < deadline
20:            admQueries  $\leftarrow$  admit the query
21:          end if
22:        end for
23:      end if
24:    end for
25:    if query is not admitted
26:      reject query with detailed rejection reasons
27:    end if
28:  end if
29:end for
```

There are two possible scenarios where queries can be admitted. **Scheduling Scenario 1:** as we have considered in previous work [11], queries have sufficient budgets and deadlines, which allows queries to be executed by processing the entire datasets. **Scheduling Scenario 2:** users have limited budgets to spend and tight deadlines to obtain data analytics results. In this way, traditional data processing on the original datasets cannot fulfill QoS requirements for user queries. To face the challenge, data splitting techniques are utilized to obtain samples of the original datasets that enable quicker processing of queries on smaller datasets with reduced costs. After admission control, if queries cannot be admitted by the AaaS platform, the reasons for rejection are presented to users to give them the chance to modify their query specifications for resubmission.

The admission control algorithm works in the following way, as shown in Algorithm 1. It gets online queries submitted by users. Then, it iteratively makes admission decisions for each query. The admission control algorithm first checks whether the BDAA requested by the query is available and the dataset is accessible. If the conditions are satisfied, the admission controller further checks whether the QoS requirements of a query can be satisfied by any resource configuration by searching all possible Resource Configurations (resConfigs) in the resource registry. The Scenario 1 of the admission control is shown in Lines 3-11. For each resource configuration, it calculates the Estimated Execution Time (EET) and Estimated Cost (EC) to execute the query on such configuration at the earliest time without queuing the query. If such a configuration exists, the query is admitted.

If a query cannot be admitted due to limitations of budgets and/or deadlines, it is then further considered for data splitting under the condition that the accuracy of the queries can be traded-off for quicker times and less costs, as shown in Lines 12-24. For each approximate query, admission control is conducted to estimate whether applying a data splitting method can satisfy its QoS requirements. The admission controller re-estimates the overall data processing cost as a result of the Data Splitting Cost (DSC) and the Data Processing Cost (DPC) while the overall data processing time as a result of the Data Splitting Time (DST) and the Data Processing Time (DPT). If a query can be executed by approximately processing smaller split datasets, it is admitted to the AaaS platform as shown in Scenario 2. After the data splitting-based admission control, all queries that cannot be admitted are given rejection reasons, as shown in Lines 25-27.

4.3.2 Resource Scheduling

In this section, we propose the profit optimization algorithm based on the ILP formulation of the optimization resource scheduling problem. We compare the performance of the optimization algorithm against several heuristic algorithms. We also propose the data splitting-based profit optimization algorithm to enable big data processing under limited budgets and tight deadlines.

1. Profit Optimization (PO) Scheduling Algorithm

We build the PO resource scheduling algorithm, as shown in Algorithm 2, to maximize profit for the AaaS platform while processing queries with SLA guarantees and minimized query processing times. PO first gets all submitted *queries* for each BDAA and makes query admission control decisions, as shown in Lines 1-3. Afterwards, it checks whether any resource configuration can process the admitted queries, *admQueries*, while satisfying QoS requirements. If such configurations are available, PO then selects the cost minimized configurations as the *resources* to execute queries. For *admQueries*, PO uses a heuristic method *selectResourceHeuristic* to decide the resource inputs, *heuResources*, for the *POScheduling* method (Lines 4-6). The *POScheduling* method adopts the ILP formulation of the resource scheduling problem with Objective Z subject to Constraints (1) - (12). For extremely large query arrival rates, if no feasible solution is returned by *POScheduling* within a scheduling timeout, the heuristic algorithm is utilized to avoid deadline violations (Lines 8-14). After adopting the scheduling solution, PO scales resources down by periodic checks. During a check, if there is any resource that is idle and reaching the end of its billing period, the PO scheduling algorithm terminates such resources to reduce resource costs (Lines 15-18).

The *selectResourceHeuristic* method (Lines 19-26) significantly reduces the running time of *POScheduling* by reducing the search space to allow it to efficiently generate scheduling solutions. It first selects the available resources, *avaResources*, from existing resources; then, it selects resources that can execute at least one query satisfying its QoS requirements from *avaResources*, as *exeResources*; afterwards, it utilizes *maximumDelayTime*-based scheduling method to maps *admQueries* to *exeResource*. The *selectResourceHeuristic* method combines *exeResources* and *newResources* as the *heuResources* to input to the *POScheduling* method.

Algorithm 2: PO Resource Scheduling Algorithm**Input:** submitted queries, resource configurations, existing resources, and BDAA**Output:** scheduling solution

```

1: for each BDAA  $\in$  requested BDAA
2:   queries  $\leftarrow$  get all submitted queries of BDAA
3:   admQueries  $\leftarrow$  admissionControl (queries, resConfigs, BDAA)
4:   if (admQueries > 0)
5:     resources  $\leftarrow$  get existing resources of BDAA
6:     heuResources  $\leftarrow$  selectResourceHeuristic (resources, resConfigs,
7:   admQueries, BDAA)
8:     solution  $\leftarrow$  POScheduling (admQueries, heuResources, BDAA)
9:     if feasible solution is returned within scheduling timeout
10:      adopt scheduling solution to execute queries
11:    end if
12:  else
13:    solution  $\leftarrow$  maximumDelayTime (heuResources, admQueries, BDAA)
14:    adopt scheduling solution to execute queries
15:  end else
16:  resources  $\leftarrow$  get existing resources of BDAA
17:  scaleDown (resources)
18: end if
19: Procedure: selectResourceHeuristic (resources, resConfigs, admQueries,
20: BDAA)
21:   avaResources  $\leftarrow$  get available resources
22:   exeResources  $\leftarrow$  select resources from avaResources
23:   maximumDelayTime (admQueries, exeResources, BDAA)
24:   if any query not scheduled in admQueries
25:     newResources  $\leftarrow$  newResourceHeuristic (resConfigs, admQueries, BDAA)
26:     heuResources  $\leftarrow$  exeResources + newResources
27:   end if
28: Procedure: maximumDelayTime (admQueries, heuResources, BDAA)
29:   sort admQueries based on MDT
30:   schedule admQueries to resources with minimal EST
31: Procedure: newResourceHeuristic (resConfigs, admQueries, BDAA)
32:   modifications  $\leftarrow$  get all possible modifications
33:   newResources  $\leftarrow$  searchHeuristic (resConfigs, modifications, admQueries,
34: BDAA)
35: Procedure: POScheduling (admQueries, heuResources, BDAA)
36:   getUpdatedInfo (admQueries, heuResources, BDAA)
37:   defineScheduleObj (Objective Z)
38:   defineScheduleConstraints (Constraints 1-12)
39:   defineTimeOut (timeout)

```

```

38:   solveScheduleProblem ()
39:   solution ← getScheduleSolution ()

```

The *heuResources* provisions sufficient resource capacity to allow *POScheduling* to schedule queries with SLA guarantees. The capacity provided by *heuResources* is close to the optimal resource configuration that can be found by *POScheduling*, which greatly reduces the search space to find the optimal resource configuration and allows *POScheduling* to return timely profit optimization resource scheduling solutions.

The ***maximumDelayTime*** method (Lines 27-29) adopts a Maximum Delay Time (MDT) based scheduling method. MDT is the maximum time a query can be delayed for execution, which is the difference between query deadline and its execution time. The *maximumDelayTime* method first sorts queries in the MDT ascending order. Then, it keeps mapping queries with minimal MDT first to resources that can satisfy their SLAs and give them the Earliest Starting Time (EST). After the MDT-based scheduling, if there is any query not scheduled due to limited resource capacity, the *newResourceHeuristic* method is utilized to create *newResources* to execute queries.

The ***newResourceHeuristic*** method (Lines 30-32) applies a search heuristic, *searchHeuristic*, to find the local cost-minimal resource configuration to create new resources for queries. The set of all possible resource configurations for a requested BDAA is represented by a directed acyclic graph, $Configuration = (Node, Edge)$, where *Node* is a set of nodes while *Edge* is a set of edges. A node is a possible resource configuration_{*i*}. An edge (*configuration_i*, *configuration_{i+1}*) indicates by applying a *modification*, *configuration_{i+1}* can be obtained from *configuration_i*. The number of resource types that are available in the AaaS platform determines the number of possible *modifications*, which include: adding the cheapest resource, adding more expensive resource, until adding the most expensive resource. The *newResourceHeuristic* method gets all possible *modifications* and iteratively applies each *modification* on current resource configuration to generate a new configuration to execute queries. It continues these steps until it finds the cost-minimal local-optimal resource configuration, *newResources*, to execute queries with guaranteed SLAs.

The ***ProfitOptimizationScheduling*** method (Lines 33-39) first gets updated information of *heuResources*, *admQueries*, and BDAA. Afterwards, it creates the ILP

solver with defined Objective Z and defined Constraints (1) - (12). Afterwards, it specifies a timeout as the maximum time that the ILP solver can run. Then *POScheduling* utilizes the ILP solver to find the optimal solution for the scheduling problem. If a feasible solution is found, it is then applied in the AaaS platform to provision resources and schedule queries. Otherwise, the MDT-based scheduling method is utilized to schedule *admQueries* to *heuResources*.

2. Heuristic Greedy Search (HGS) Algorithm

The HGS scheduling algorithm is a combination of the *selectResourceHeuristic* method and the *maximumDelayTime* method. After admitting queries, the HGS scheduling algorithm first finds *heuResources* by applying the *selectResourceHeuristic* method and heuristically schedules queries to *heuResources* by applying the *maximumDelayTime* scheduling method. If the *POScheduling* method of the PO scheduling algorithm cannot return any feasible scheduling solution with defined timeout, the *maximumDelayTime* method is utilized as the scheduling method for PO to schedule queries in a heuristic way to avoid deadline violations. In this particular scenario, the scheduling solution generated by the PO scheduling algorithm is the same as the HGS scheduling algorithm.

3. Adaptive ILP (AILP) Algorithm

The AILP scheduling algorithm [6] is a two-phase resource scheduling algorithm, which aims to provide cost-effective resource scheduling solution to execute queries with SLA guarantees on budgets and deadlines. AILP provisions a scalable resource scheduling solution by applying a scheduling method based on its ILP modeling for two scheduling phases. In Phase 1, AILP aims to maximize resource utilization of existing resources and scale resources down by terminating existing resources when resources are over-provisioned to minimize resource cost with SLA guarantees. In Phase 2, if there is any query not successfully scheduled due to the under-provision of resources, AILP scales resources up to execute queries by creating resources with minimal resource cost while satisfying SLAs. To avoid unpredicted arrival of large number of queries, AILP specifies timeout to limit algorithm running time as latest time ILP-based scheduling method can run. If the timeout is exceeded when no feasible solution is returned, AILP utilizes *maximumDelayTime* scheduling method to make scheduling decision to avoid deadline violations. If the ILP-based scheduling method of AILP cannot return feasible

solution, AILP generates the same solution as the solution generated by running AGS scheduling algorithm directly in this particular scenario.

4. Adaptive Greedy Search (AGS) Algorithm

The AGS scheduling algorithm [6] is a two-phase scheduling heuristic, which applies *maximumDelayTime* and *searchHeuristic* methods. After admission control, AGS schedules *admQueries* to existing resources by applying *maximumDelayTime* scheduling method in Phase 1. If created resources have limited capacity to execute queries satisfying QoS requirements, AGS then creates new VMs to process queries in Phase 2 by applying the *searchHeuristic* method to find local cost-minimal configuration of resources, *newResource*, and schedules queries to *newResource* based on the *newResourceHeuristic* method by applying a local cost-minimal resource configuration.

5. Data Splitting-Based Profit Optimization (SPO) Scheduling Algorithm

For big data analytics with limited budgets or tight deadlines, such analytics requests cannot always be admitted due to QoS constraints. To tackle this challenge, data splitting-based scheduling algorithms can be applied to process a subset of the large datasets and return analytics solutions with less costs and faster responses.

The SPO scheduling algorithm applies SLA and data splitting based scheduling that allocates resources to execute queries on smaller split data samples with SLA guarantees. The SPO scheduling algorithm enables processing queries with tight deadlines and limited budgets and allows the analytics results to be returned in a timely manner with controllable costs. It also benefits users who have cost-saving purposes and want timely results where accuracy is not a limited factor for the tasks. The pseudo code of the SPO algorithm is shown in Algorithm 3.

Algorithm 3: SPO Resource Scheduling Algorithm

Input: submitted queries, resource configurations, existing resources, and BDAA

Output: scheduling solution

```

1: for each BDAA  $\in$  requested BDAA
2:   queries  $\leftarrow$  get all submitted queries of BDAA
3:   admQueries  $\leftarrow$  admissionControl (queries, resourceConfigurations, BDAA)
4:   apprQueries  $\leftarrow$  get approximate queries
5:   for each query in approximateQueries
6:     dataSplittingTasks  $\leftarrow$  generate data splitting tasks
7:   end for

```

```

8:  anaTasks  $\leftarrow$  admQueries + dataSplittingTasks
9:  if (anaTasks > 0)
10:    resources  $\leftarrow$  get existing resources of BDAA
11:    heuResources  $\leftarrow$  selectResourceHeuristic (resources, resConfigs, anaTasks,
BDAA)
12:    solution  $\leftarrow$  SPOScheduling (anaTasks, heuResources, BDAA)
13:    if feasible solution is returned within scheduling timeout
14:      adopt scheduling solution to execute queries
15:    end if
16:    else
17:      solution  $\leftarrow$  maximumDelayTime (heuResources, anaTasks, BDAA)
18:      adopt scheduling solution to execute queries
19:    end else
20:    resources  $\leftarrow$  get existing resources of BDAA
21:    scaleDown (resources)
22:  end if
23: end for
24: Procedure: SPOScheduling (anaTasks, heuResources, BDAA)
25:  getUpdatedInfo (anaTasks, heuResources, BDAA)
26:  defineScheduleObj (Objective Z)
27:  defineScheduleConstraints (Constraints 1-14)
28:  defineTimeOut (timeout)
29:  solveScheduleProblem ()
30:  solution  $\leftarrow$  getScheduleSolution ()

```

The scheduling scenario for SPO works as follows: users submit data analytics requests to the AaaS platform requesting BDAAAs to process their datasets. For each BDAA, the SPO scheduler firstly gets all the queries, resources running the BDAA, and the current resource configuration of the AaaS platform as input data. The resource scheduler then calls the admission controller to admit queries. After admission control, SPO schedule queries based on the SPOScheduling method, which adopts the ILP formulation of the resource scheduling problem with Objective Z subject to Constraints (1)-(25).

If there are queries that cannot be admitted due to constraints of budgets and deadlines to process the large datasets, the Admission Controller checks among these queries and tries to re-admit the approximate queries, *apprQueries*, which allow approximate processing. If applying the splitting technique leads to satisfactory service delivery with respect to QoS requirements of queries, such queries are admitted. Afterwards, SPO scheduler generates the data analytics tasks, *anaTasks*, including data splitting tasks and

data processing tasks, as shown in Lines 1-8. SPO then selects `heuResources` by applying `selectResourceHeuristic` method. Afterwards, the SPO scheduler applies `SPOScheduling` to generate profit-optimal scheduling solution, as shown in Lines 9-12.

The ***SPOScheduling*** method (Lines 24-30) first gets the up-to-date information regarding the BDAA, queries, and resources as input to the optimization solver. Then, SPO specifies the objective and constraints of the optimization scheduling problem. SPO shares the same objective function as the PO scheduling algorithm while accommodating additional constraints for data splitting and data dependencies based scheduling. Finally, SPO utilizes the solver to generate the optimized scheduling solution. If feasible solutions are generated within scheduling timeout, the solution is applied to execute queries. Otherwise, the heuristic scheduling approach `maximumDelayTime` is utilized to generate alternative solution to schedule resources (Lines 13-19). After applying the scheduling solution to execute queries, the SPO scheduler further conducts periodic check to scale resources down (Lines 20-23). After the current schedule, new periodic schedules are triggered as more online queries submitted by users arrive at the AaaS platform and need to be scheduled and processed with SLA guarantees.

4.4 Performance Evaluation

The admission control and profit optimization scheduling algorithms aim to maximize the profit for AaaS platforms and deliver AaaS services with SLA guarantees. In this section, we conduct extensive experiments to evaluate the effectiveness and efficiency of the proposed algorithms in admission control, SLA guarantee, profit enhancement, cost saving, resource configuration, algorithm running time, and query admission.

4.4.1 Experiment Setup

1. Experiment Environment

To evaluate the proposed admission control and resource scheduling algorithms, we build the AaaS framework in CloudSim [122], which is a discrete event simulation platform that enables modeling and simulation of cloud environments with on-demand resource provisions and configurations. It allows us to conduct repeatable and controllable evaluations of our proposed resource scheduling solutions without incurring expensive IaaS resource and BDAA costs. We utilize IBM ILOG CPLEX 5.5 as the ILP

solver. CPLEX is an optimization software package, which is an analytics decision support toolkit for rapid development and deployment of optimization models using mathematical and constraint programming [134]. We study algorithm performance for different scheduling scenarios including periodic scheduling that schedules queries for each Scheduling Interval (SI) and non-periodic (realtime) scheduling that schedules queries based on arrivals.

Table 4.2: Query Admission Rates and SLA Guarantees

Rate	Real Time	Periodic					
		10	20	30	40	50	60
Admission	84.0	79.3	74.8	71.8	68.5	65.3	63.0
SLA Guarantee	100	100	100	100	100	100	100

2. Resource Configuration

We simulate a datacenter that consists of 500 physical nodes. Each node has 50 CPU cores, 100GB memory, 10TB storage, and 10GB/s bandwidth. We simulate five types of memory optimized VMs based on the Amazon EC2 VM model. The VMs are r3.large, r3.xlarge, r3.2xlarge, r3.4xlarge, and r3.8xlarge [123]. The resource configuration to execute queries can be a single VM or a cluster of VMs. A resource is charged based on the hourly utilization. The unit for memory is GiB, for storage is GB, for the resource cost is dollar/hour, and for the SI is minute.

3. Data Analytics Workload

The data analytics workload generated in this study is based on Big Data Benchmark [124] and a Normal Distribution (ND) based data splitting work [57]. The ND based data splitting technique is utilized to split the original large datasets to obtain smaller split datasets to process, which provides significant performance improvement compared to process the original datasets. The ND-splitting work details the resource configurations and times to split large datasets using ND based data splitting applications while, the Big Data Benchmark details query processing times and the resource configurations using the Amazon EC2 Cloud to run queries on large datasets using different data analytic frameworks with given data size, data type, and data location. Based on the data splitting and processing information, we model the time for data processing and splitting on BDAAAs under different resource configurations.

Each query request contains the following information: *Submission time* is generated using a Poisson process with 1 minute mean Poisson arrival interval. *Query type* includes scan query, aggregation query, join query, and User Defined Function (UDF) query. *BDAA*s: 4 types of BDAA are considered in our experiment that are built on Impala (disk) (BDAA₁) [132], Shark (disk) (BDAA₂) [133], Hive (BDAA₃) [67], and Tez (BDAA₄) [69]. *Resource time* is modeled based on the resource requirements of queries.

As the performance of queries can vary, we model 10% performance variation by introducing a variation coefficient [125], which is generated from a Uniform Distribution with upper bound 1.1 and lower bound 0.9. *User*: we simulate 50 users who submit queries and request for BDAA. *Deadline* is generated as the benchmark does not contain QoS requirements on deadline, so is budget. Two types of deadline are considered, tight deadline and loose deadline. Tight deadline is generated using a Normal Distribution (3, 1.4). Loose deadline is generated similarly using a Normal Distribution (8, 3) [126]. Similarly, *Budget* contains tight budget and loose budget. Tight budget is generated using a Normal Distribution (3, 1.4) while loose budget is generated using a Normal Distribution (8, 3).

4.4.2 SLA Based Resource Scheduling

We generate an approximate 7 hours query workload on 30GB data that contains 400 queries. We conduct the realtime scheduling and periodic scheduling by varying the SI from 10 to 60 (unit: minute) to evaluate the algorithm performance regarding admission control, SLA guarantee, profit enhancement, resource cost, resource configuration, and algorithm runtime.

1. Admission Control and SLA Guarantees

We conduct the realtime scheduling and periodic scheduling by varying the SI from 10 to 60 (unit: minute) to evaluate the algorithm performance regarding admission control and SLA guarantee. We calculate the query admission rate and SLA guarantee rate based on the obtained results, as shown in Table 4.2. For realtime scheduling, the admission rate is 84.0%. For periodic scheduling with SI varied from 10 to 60, the admission rate decreases from 79.3% to 63.0% since the longer SI is, the more queries are rejected due to deadline constraints. Thus, shorter SIs are recommended as more queries can be admitted to increase user satisfaction and enlarge market share. All admitted queries are successfully executed with 100% SLA guarantees, which is the

result of effective and efficient admission control and resource scheduling. Service delivery with SLA guarantees helps to increase user satisfaction and reputation of AaaS providers.

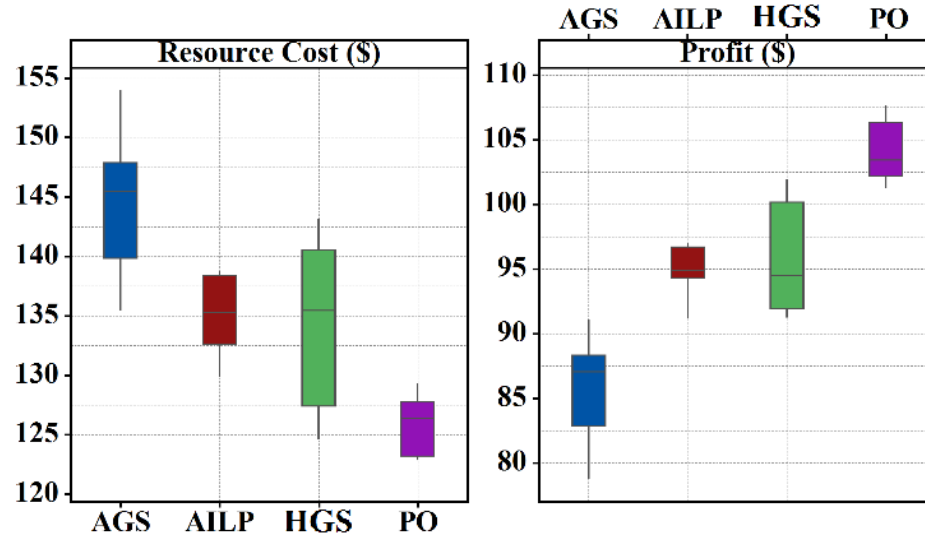


Figure 4.3: Median Profits and Resource Costs of PO, HGS, AILP, and AGS Algorithms.

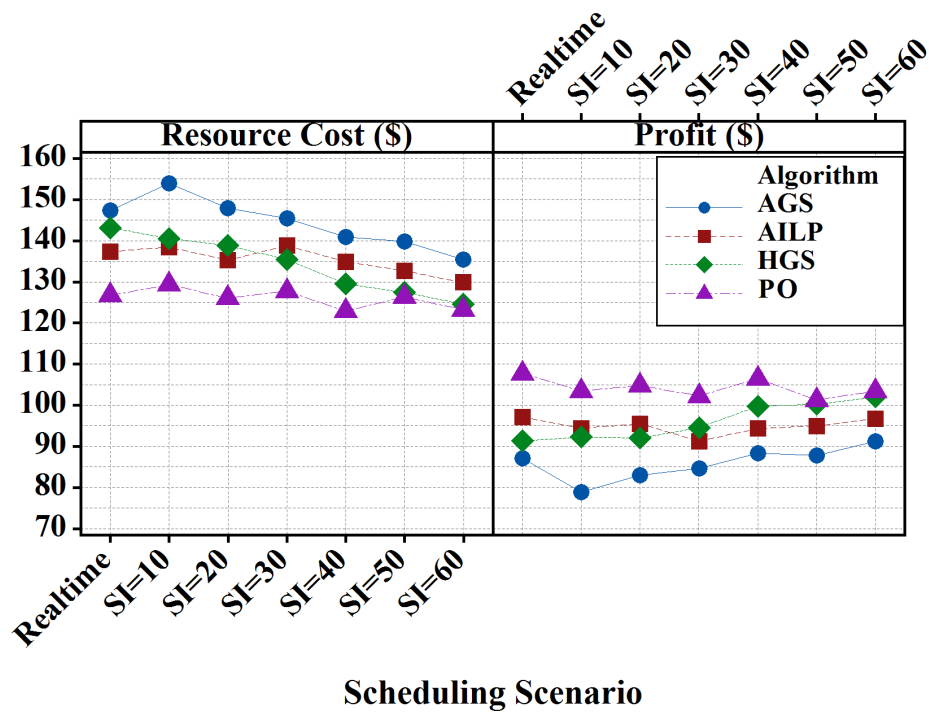


Figure 4.4: Profits and Resource Costs of PO, HGS, AGS, and AILP Algorithms.

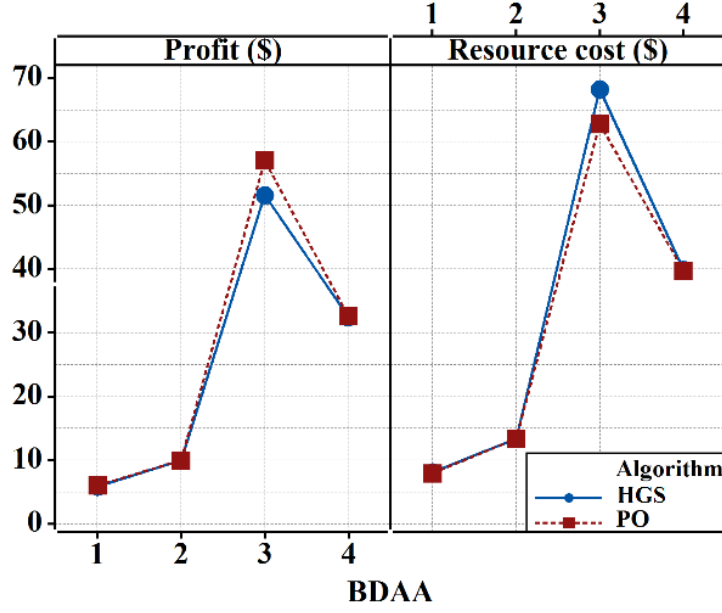


Figure 4.5: Profits and Resource Costs of Different BDAA.

2. Profit Enhancement and Cost Saving

We evaluate the mean and median resource costs and profits of PO compared to HGS, AILP, and AGS for all scheduling scenarios, as shown in Figure 4.3. Compared to HGS, AILP and AGS, PO saves 7% to 15% more resource costs respectively while creating 9% to 19% higher profit. Moreover, the mean profit created by PO is \$104 and the resource cost is \$126. Compared to HGS, AILP, and AGS, the profit of PO is [9%, 10%, 21%] higher while the resource cost is [7%, 7%, 15%] less. The above results indicate the overall performance advantages of PO in achieving higher cost saving and profit enhancement objectives.

1) Profits and Resource Costs for Different Scenarios

We detail the performance evaluation of PO in different scheduling scenarios, as shown in Figure 4.4. For realtime and periodic scheduling, results indicate that PO creates the highest profit and generates the least cost while AGS performs the opposite. As an optimization algorithm, PO performs significantly better than AILP to generate profit optimization solutions. As a heuristic algorithm, HGS performs significantly better than AGS that serves as a suitable alternative algorithm to deliver cost-effective scheduling solutions for extreme large-scale arrivals of queries. From this section on, we do not include AGS and AILP in the evaluation for simplicity purposes.

The obtained results also show that PO creates higher profit and generates less resource cost compared to HGS for both realtime and periodic scheduling. As SI

increases, due to high query arrival rates, the scheduling decisions generated by PO for some schedules may not be the optimal ones, which weaken the advantages of PO.

SI can be adjusted to a suitable scale based on the arrival rate of queries for different workloads to enable PO to generate timely profit optimization solutions. To achieve higher cost saving and profit enhancement, while admitting more queries, periodic scheduling with SI=40 is the best solution to execute the query workload.

2) Performance Evaluation for Different BDAAs

We also evaluate the efficiency of PO for each BDAA with SI=40. As shown in Figure 4.5, for different BDAAs, the resource cost and profit vary significantly. This is caused by variations in the number of admitted queries and their dynamic needs for resources. The results show that PO creates higher profits and generates less resource costs for BDAA₁, BDAA₃, and BDAA₄ compared to HGS. For BDAA₂, the profit and resource cost of PO and HGS are the same. We further analyze the results, which show it takes 15.8 hours for the PO scheduling algorithm to execute queries on 7 * r3.large VMs while it takes 12.4 hours for HGS to execute queries on 7 * r3.large and 1 * r3.xlarge VMs. If PO and HGS are used to run a different workload, the performance advantages of the PO scheduling algorithm for BDAA₂ can be shown.

Table 4.3: Resource Configuration

Scheduling Scenario		HGS	PO
Real Time		56 * r3.large	36 * r3.large
Periodic	<i>SI=10</i>	50 * r3.large	33 * r3.large
	<i>SI=20</i>	42 * r3.large	30 * r3.large
	<i>SI=30</i>	37 * r3.large 1 * r3.xlarge	25 * r3.large 1 * r3.xlarge
	<i>SI=40</i>	32 * r3.large, 1 * r3.xlarge	26 * r3.large
	<i>SI=50</i>	27 * r3.large 2 * r3.xlarge	22 * r3.large, 2 * r3.xlarge
	<i>SI=60</i>	25 * r3.large, 1 * r3.xlarge	19 * r3.large, 1 * r3.xlarge

3) Performance Evaluation for Single Schedules

We further conduct experiments to evaluate advantages of the PO in single schedule without accumulated results from previous schedules. As shown in Figure 4.6, for different SIs, the profit created by PO is higher while the resource cost is lower than

HGS. We notice a trend in profit enhancement with the increase of SIs as more queries are admitted and PO and HGS gain better knowledge to make scheduling decisions.

Experiment results show that PO can generate less resource cost and create higher profit not only for single schedules, but also for each BDAA and the entire query workload for all scheduling scenarios with detailed analysis shown. In summary, we conclude that PO is the best profit optimization algorithm for the AaaS platform compared to HGS, AILP, and AGS for Scheduling Scenario 1.

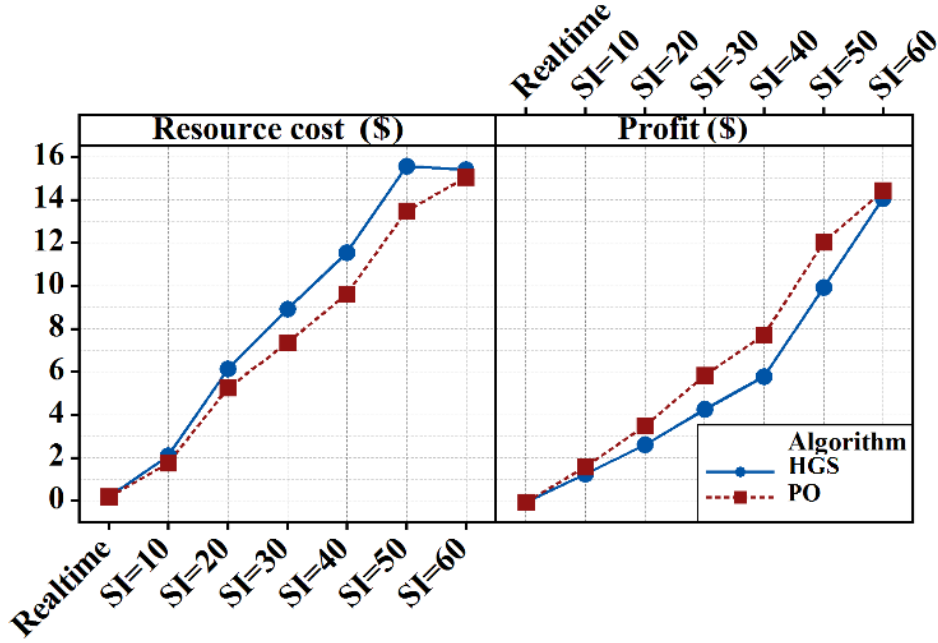


Figure 4.6: Profits and Resource Costs of PO and HGS Algorithms for Single Schedules.

3. Resource Configuration

The resource configuration to execute the query workload of PO and HGS is shown in Table 4.3. The results show that, for all scheduling scenarios, PO can efficiently reduce the numbers of VMs. This is because that PO can make full utilization of the capacity of created resources. To achieve this, PO gives priority to utilize earlier created VMs, which maximizes their utilization and helps to reduce the load in latter created VMs for VM termination during periodic checks. Creating more VMs will cause more delays [128] to serve requests that may cause deadline violations and higher costs. Thus, PO better serves our objectives of SLA guarantee and cost saving.

4. Algorithm Running Time (ART)

We record the ART for PO, HGS, AILP and AGS to execute the query workload, as shown in Figure 4.7. For realtime scheduling, a small number of queries are scheduled based on their arrivals. Thus, ART is not a limiting factor for all algorithms since it only takes a few seconds to schedule queries. For periodic scheduling, it takes few seconds for AGS and HGS to schedule queries for all SIs. For AILP, ART increases rapidly with the increase of SI, which tends to reach a constant value, which is the timeout to run the algorithm. For PO, ART is greatly reduced compared to AILP. For 1 hour scheduling intervals, it takes an average 237 seconds to finish scheduling, which is sufficient fast to solve the scheduling problem. Thus, ART is not a limiting factor for PO to serve as the scheduling algorithm for the AaaS platform to solve the profit optimization problem.

In summary, we conclude that PO is the best profit optimization algorithm of the AaaS platform compared to HGS, AILP, and AGS for Scheduling Scenario 1, which can use less resources, generate less resource cost and create higher profit not only for each schedule, but also for each BDAA and the entire query workload for all realtime and periodic scheduling.

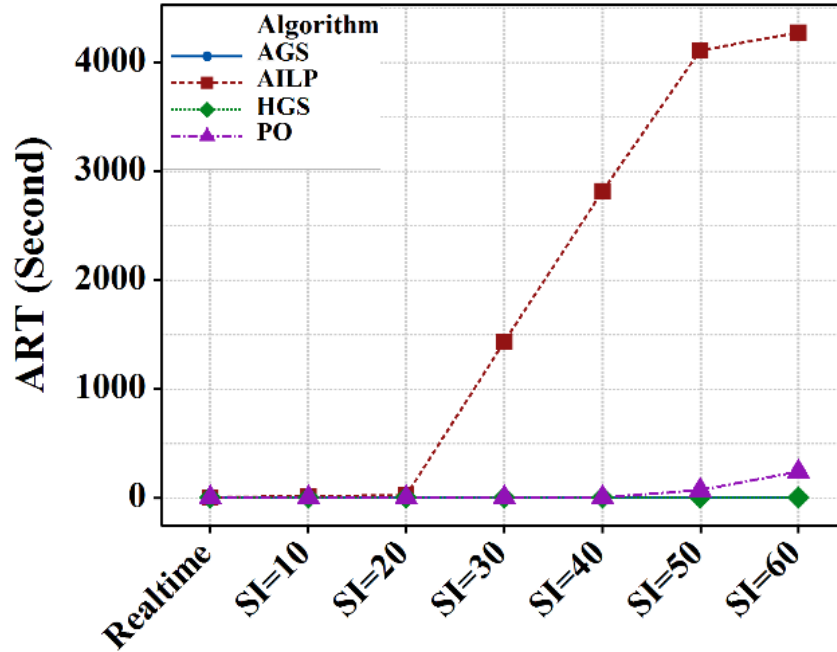


Figure 4.7: ART of PO, HGS, AILP, and AGS Algorithms.

4.4.3 SLA and Data Splitting Based Resource Scheduling

To overcome the limits of Scheduling Scenario 2 that users have tight deadline and budget constraints of queries, data splitting and sampling are utilized to enable approximate query processing to trade-off accuracy for faster response times and reduced resource costs. It is noted that the SPO algorithm serves general purpose by supporting different data splitting and sampling techniques. To illustrate the efficiency and effectiveness of our proposed SPO algorithm, we apply a normal distribution based data splitting method based on our previous work that supports splitting-based big data processing [57]. The ND splitting technique can be replaced by other efficient data processing techniques as long as we can profile the applications to know resource configurations and processing times of the data splitting and sampling methods so that the resource scheduler can make the SLA-based profit optimization decisions. We will continue investigating efficient data splitting and sampling techniques to support the AaaS platforms as part of the future work.

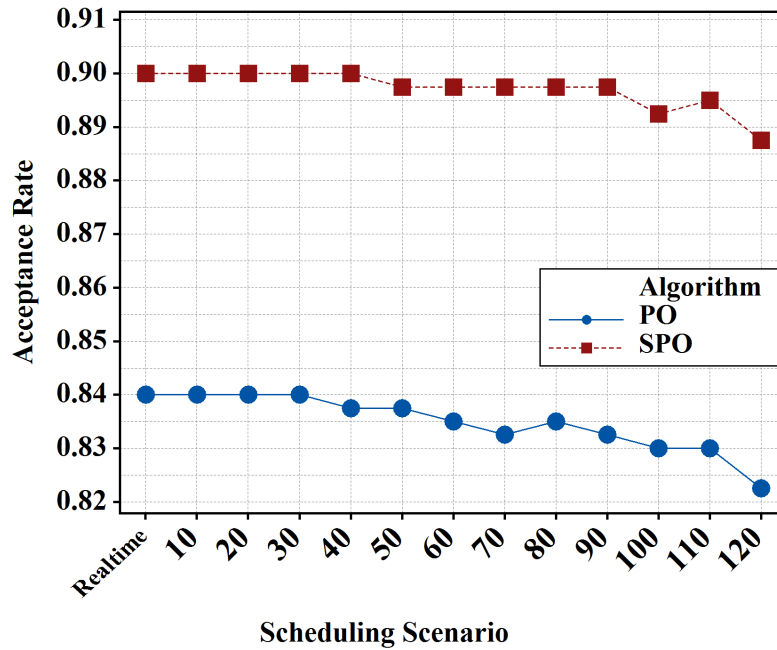


Figure 4.8: Admission Rates of PO and SPO Algorithms.

The ND-based splitting technique works on normal distributed or near normal distributed datasets that contain at least one numeric column and more numeric columns. It applies a ND based splitting algorithm to generate split datasets as data representatives to improve the accuracy of data analytics. The ND-based method can be efficiently

applied to data analysis for medical datasets and enable AaaS platforms to return query results based on a single representative data sample. The representative character of ND-based datasets is evidenced by various medical datasets such as birth weight distribution and body mass index. Applying ND-based splitting to analyze data representatives can provide relatively more comprehensive and accurate information than randomly splitting. The inclusive character of ND-based splits datasets can improve time efficiency, reduce resource costs, and improve accuracy of big data analytics.

Table 4.4: Query Admission and SLA Guarantee

Rate	BDAA ₁	BDAA ₂	BDAA ₃	BDAA ₄
SPO	65.3	98.9	99.0	95.2
PO	65.3	89.5	89.2	89.5

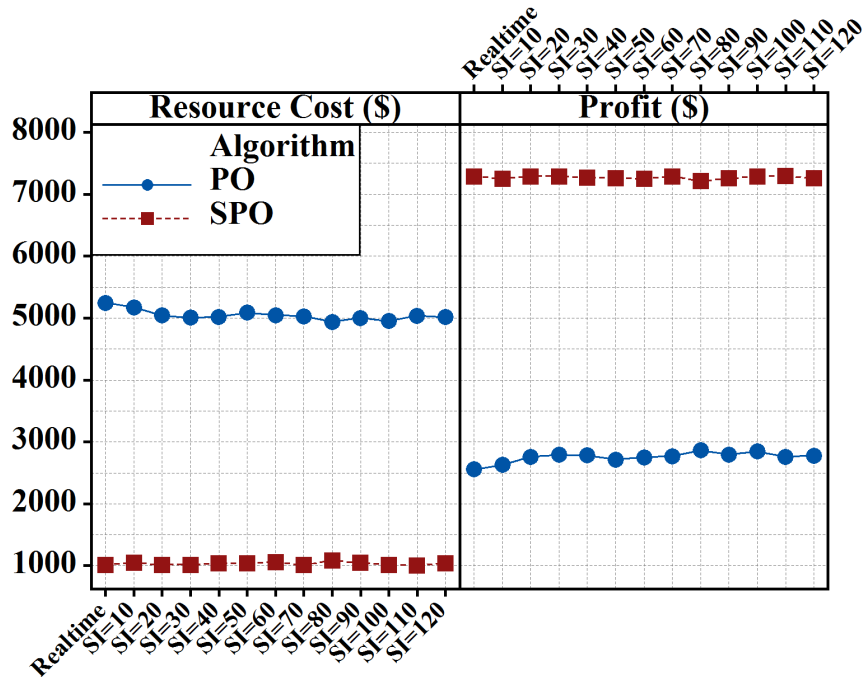


Figure 4.9: Resource Costs and Profits of PO and SPO Algorithms.

The datasets utilized for ND based scheduling are health datasets, which contain the patient information and their hospital records provided by the online sources with records including patients' marriage status, in-hospital duration [57]. The original health

datasets are not sufficiently large to apply the data splitting based scheduling method to show its performance benefits. Thus, the required resources for data splitting are enlarged to a suitable scale based on the original datasets. In this study, we generate 1 TB query workload based on big data benchmark with splitting time of health data source as the experimental data. We conduct the following experiments to evaluate the performance of data splitting based resource scheduling and non data-splitting based resource scheduling.

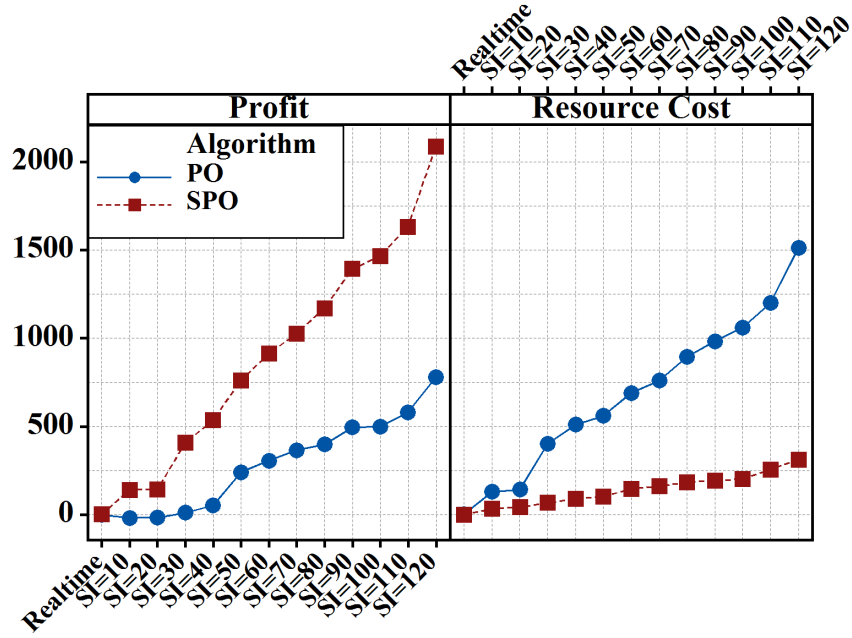


Figure 4.10: Profits and Resource Costs of PO and SPO Algorithms for Single Schedules.

1. Admission Control and SLA Guarantees

As indicated in Figure 4.8, the admission rates of SPO for different scheduling scenarios are in the range of [88%, 90%], which is approximately 6% higher compared to the admission rates of PO in the range of [82%, 84%]. The fluctuation of the admission rates is caused by the Poisson arrival of the queries. The results show overall significantly increased query admission rates. The reason is that data-splitting based resource scheduling enables the AaaS platform to admit queries whose analytics solutions cannot be generated due to tight deadlines and budgets constraints. Higher query admission rates bring more benefits to the AaaS platform with regards to market share enlargement and profit enhancement.

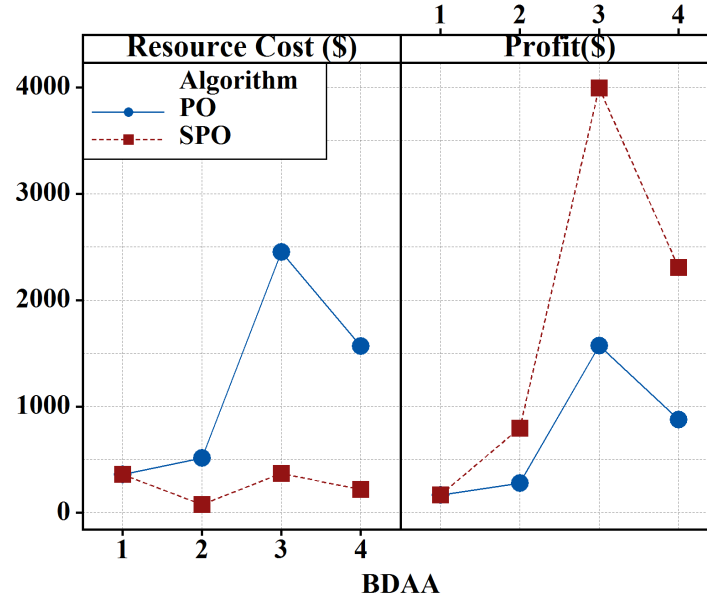


Figure 4.11: Profits and Resource Costs of PO and SPO Algorithms for BDAA with SI=80.

Table 4.5: Resource Configuration

Scheduling Scenario	PO	SPO
Realtime	63 * r3.large	75 * r3.large
SI=10	59 * r3.large	72 * r3.large
SI=20	57 * r3.large	66 * r3.large
SI=30	60 * r3.large	63 * r3.large
SI=40	53 * r3.large	58 * r3.large
SI=50	52 * r3.large	59 * r3.large
SI=60	51 * r3.large	55 * r3.large
SI=70	50 * r3.large, 1 * r3.xlarge	53 * r3.large, 1 * r3.xlarge
SI=80	49 * r3.large	53 * r3.large
SI=90	49 * r3.large, 1 * r3.xlarge	52 * r3.large, 1 * r3.xlarge
SI=100	43 * r3.large, 2 * r3.xlarge	46 * r3.large, 2 * r3.xlarge
SI=110	41 * r3.large,	45 * r3.large,
SI=120	1 * r3.xlarge	1 * r3.xlarge

2. Profit Enhancement and Cost Saving

As shown in Figure 4.9, we analyze the profit enhancement and cost saving, which are the key performance indicators of scheduling algorithms SPO and PO.

1) Average Profits and Resource Costs

With a 6% higher acceptance rate, SPO creates an average profit of \$7272, which is 2.6 times that of PO with an average resource cost of \$1033, which is 20% that of PO. The results indicate a significant profit enhancement and cost saving advantage of applying SPO. The reason is that data splitting based scheduling leads to higher query admission rate for higher query incomes as well as a lower resource costs by processing data samples, which subsequently lead to a significant enhancement of profit.

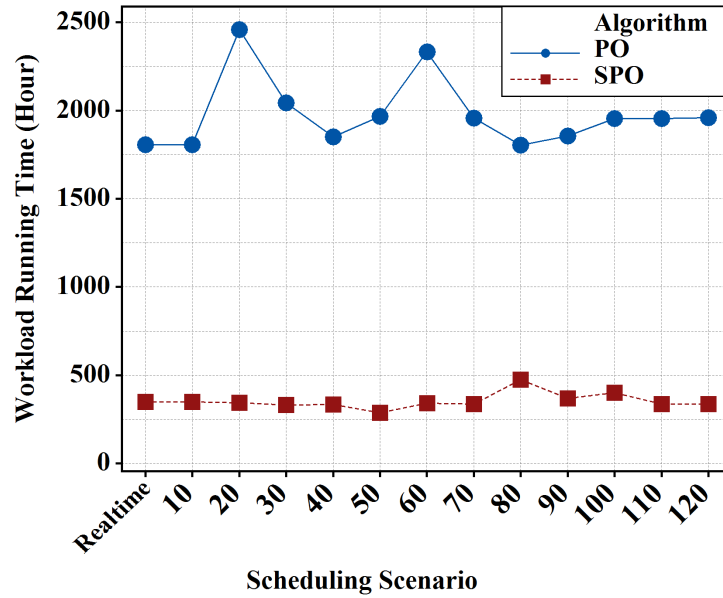


Figure 4.12: Workload Running Times of PO and SPO Algorithms.

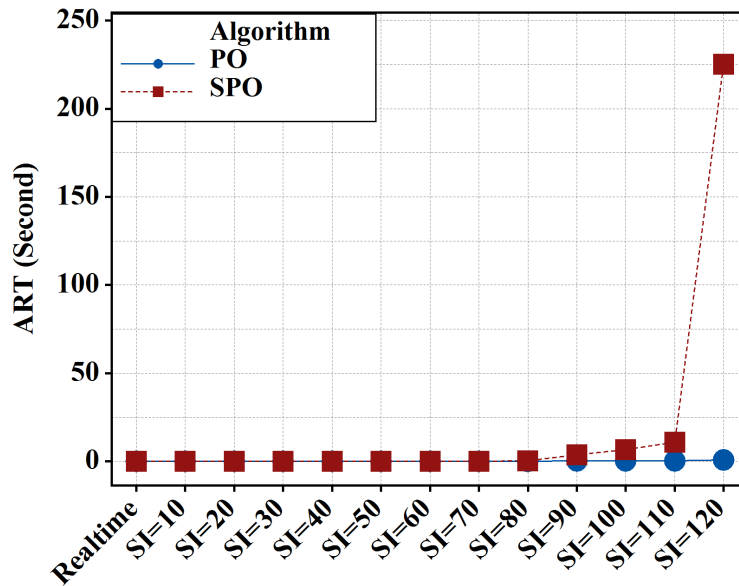


Figure 4.13: ART of PO and SPO Algorithms.

2) Profits and Costs for Different Scenarios

Results show SPO performs significantly better than PO for all scheduling scenarios including realtime scheduling and periodic scheduling with SI from 10 minutes to 2 hours, the profits generated by SPO are in the range of [\$7217, \$7351] while the profits created by PO are in the range of [\$2559, \$2863] based on the Amazon EC2 pricing model. The resource costs of PO are [\$5246, \$4941] while the resource costs of SPO are [\$1002, \$1088].

3) Performance Evaluation for Single Schedules

We further analyze the performance of PO and SPO for one schedule that is not affected by other scheduling intervals or accumulated results. As shown in Figure 4.10, we can see a more rapid increasing trend of profits created by SPO than PO while the resource costs of PO increase much higher than SPO as the scheduling interval increases from realtime to a period of 120 minutes. The results show that for the same scheduling intervals and same query arrival rates, SPO steadily outperforms PO in terms of cost saving and profit enhancement for all scheduling scenarios. We also identify that for realtime scheduling and periodic scheduling with SI of 10 and 20, the profits created by PO incur negative values that indicate that the accepted queries cannot fully utilize the cloud resources and the query income cannot cover the resource costs. Thus, fully utilizing cloud resources with appropriate SI is important for profit enhancement.

4) Performance Evaluation for Different BDAAs

We use SI=80 as the scheduling interval to analyze the performance of PO and SPO. It is noted that at SI=80, the SPO scheduling algorithm generates the lowest profits among all periodic SIs while the PO scheduling algorithm creates the highest profits among all periodic SIs. As shown in Figure 4.11, since BDAA₁ does not enable approximate query processing due to its accuracy requirements, the profit created by SPO is equal to the profit created by PO. For BDAA₂, BDAA₃, and BDAA₄, approximate query processing is allowed to enable quicker responses and less resource costs without strict accuracy requirements. The query acceptance rates of different BDAAs are shown in Table 4.4. Even for the highest profit performance of the PO scheduling algorithm while at the lowest profit performance of SPO among all periodic SIs, for BDAA₂, BDAA₃, and BDAA₄, with higher admission rates 10.6%, 11.0%, and 6.4% accordingly, the profits of SPO are 1.8, 1.9, and 1.7 times improved over PO while

the resource costs of SPO are 5.3, 6.7, and 6.5 times lower than PO. We conclude that for the other SIs, SPO can better outperform PO for cost saving and profit enhancement.

3. Resource Configuration and Tight Deadline Constraints

We identify that SPO utilizes a higher number of cloud resources for both realtime scheduling and periodic scheduling compared to PO, as shown in Table 4.5. The reason is that SPO admits a higher number of queries than PO. For queries with tight deadlines, new resources are required to execute them simultaneously to avoid deadline violations. PO and SPO both make optimal resource scheduling solutions. For PO, as a profit optimization algorithm, it has already made the optimal scheduling solution to maximize the profits, minimize query processing times, and maximize resource utilization based on relatively less admitted queries compared to SPO. For SPO, although data splitting based sampling enables the processing of smaller data samples to save resource costs whilst reducing the required resource times, the benefits of resource time deduction due to less query processing cannot beat the tight deadline constraints of query processing. Despite the fact that creating fewer resources is beneficial and preferable for AaaS platforms, SPO creates a higher number of cloud resources for SLA guarantees to meet tight deadline requirements.

4. Workload Running Time

Query workload running is the time for the AaaS platform to finish processing the entire query workload. Results indicate that SPO can finish processing the query workload within 500 hours for both realtime and periodic scheduling, as shown in Figure 4.12. However, for PO it takes from 1750 hours to 2500 hours to process the queries, which is 3 to 5 times that of SPO. The results show that SPO can significantly reduce query processing times, which better serves the goals of AaaS platforms for performance enhancements and improve user satisfaction.

5. Algorithm Running Time

For ART, as shown in Figure 4.13, we observe that both PO and SPO can produce scheduling solutions in a timely manner for both realtime and periodic scheduling with SI from 10 minutes to 120 minutes. The average ART for the PO scheduling algorithm for different SIs is between [0.011, 0.767] seconds, while the ART for the SPO scheduling algorithm is between [0.007, 225.313] seconds. We note that ART for the PO scheduling algorithm exhibits a relatively stable trend while we notice a significant increase in the average ART of the SPO scheduling algorithm for SI=120. Through

further analysis of the experiment results, we identify two scheduling intervals reached by the system-defined scheduling timeout. The returned scheduling solutions of the SPO scheduling algorithm may be sub-optimal solutions it has found thus far to fulfill SLA requirements.

For unexpected extreme high numbers of query arrivals, if there is no optimal solution found, both PO and SPO can switch to heuristic methods to generate scheduling solutions to fulfill tight deadline requirements as the heuristic algorithms that have been refined multiple times can deliver near-optimal scheduling solutions. The ART is not a limiting factor for both SPO and PO to deliver timely and satisfactory scheduling solutions for all use cases.

We focus on reducing the ART of the optimization scheduling algorithms through a refined problem formulation that introduces less variables and constraints in the ILP model with reduced input scales by applying several heuristic algorithms to input near-optimal resource configurations to PO and SPO. The reduced problem and input scale enable PO and SPO to generate timely profit optimization scheduling solutions for large query arrivals.

4.5 Related Work

Analyzing big data to find potential insights in the data is essential for individuals, organizations, and governments to make better decisions. AaaS platforms aim to deliver on-demand data analytics services to users in various domains at lower cost with SLA guarantees in cloud computing environments. Our research focuses on providing effective and efficient cloud resource scheduling algorithms to support the AaaS platform. We compare our work with most related works and detail the comparisons based on various parameters in Table 4.6.

Zhao et al. [6] propose an admission control and resource scheduling algorithm to increase profits for AaaS providers by offering a cost-effective and SLA-guaranteed resource scheduling solution. Zhao et al. [11] further propose a profit-optimization scheduling algorithm to maximize profits for the AaaS platform while minimizing query response times with SLA guarantees. This work further addresses the challenge of resource scheduling for big data processing with limited deadlines and budgets by proposing a data-splitting and task-dependency based admission control and profit optimization algorithm.

Mian et al. [74] focus on resource provisioning for data analytic workloads in a public cloud that aims to find cost-effective configuration using heuristic scheduling algorithms. Their work schedules queries for one application while allowing SLA violations to trade for a cheaper resource cost, which is different from our SLA guaranteeing perspective. Sfrent et al. [107] propose asymptotic scheduling for many tasks computing in big data platforms to maximize the number of the jobs that can be processed and do not consider SLA guarantees. Sun et al. [12] propose a general-purpose analytic framework to provision cost-effective solution for multi-tenancy scenarios. However, they do not consider admission control or address scheduling algorithms with SLA guarantees.

There are extensive works on workflow scheduling. Kllapi et al. [72] address the traditional workflow scheduling problem with time and money optimality. Zulkernine et al. [21] propose a conceptual architecture of cloud-based Analytics-as-a-Service, which does not consider admission control. Zhang et al. [96] propose multi-objective scheduling for many tasks to reduce overhead times and finding a near optimal solution.

Resource provisioning for MapReduce tasks is a well-explored research area, which targets at a specific application model and does not serve general data analytics purposes. Tang et al. [98] propose a self-adaptive scheduling algorithm to reduce the start time of the reduce task of MapReduce. Alrokayan et al. [93] adopt Lambda architecture and focus their work on the batch layer to offer cost-aware and SLA-based algorithms for scheduling MapReduce tasks. Bicer et al. [135] propose a modeling-driven resource allocation framework to support time and cost sensitive execution for MapReduce applications.

There have been several cloud enterprises delivering data analytics services. Amazon offers data analytics web services such as Elastic MapReduce [136] that is a service to enable easily and cost-effectively process of big data. Amazon charges users for the cloud resources and the managed MapReduce services. Google offers Big Query [137], which is a fully managed Database-as-a-Service for large-scale data analytics. IBM provides BigInsights [138], which is a Hadoop-as-a-Service to offers the performance and security of an on-premises deployment.

There have been a number of research works focus on leveraging Database-as-a-Service. Relational Cloud [139] is a project that investigates technologies and challenges related to Database-as-a-Service in clouds. Agarwal et al. [140] propose BlinkDB for approximate query processing on large datasets with bounded response time and

bounded errors. Narasayya et al. [45] propose SQLVM to provision multi-tenant relational Database-as-a-Service in multi-tenant Microsoft Azure SQL Database. Ahmad et al. [39] propose interaction-aware scheduling for report generation workloads to maximize database performance.

Table 4.6: Related Works

Parameter	Related Works													Our Work
	6	11	21	41	57	72	74	75	76	79	93	103	135	
<i>AaaS</i>	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	Y
<i>Cost saving</i>	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
<i>Budget</i>	Y	Y	N	N	N	Y	N	N	N	N	Y	N	Y	Y
<i>Deadline</i>	Y	Y	N	N	N	Y	N	N	N	N	Y	N	Y	Y
<i>Admission control</i>	Y	Y	N	N	N	N	N	Y	N	N	N	N	N	Y
<i>Scalable provision</i>	Y	Y	Y	N	N	N	Y	Y	N	Y	N	Y	N	Y
<i>Automatic provision</i>	Y	Y	N	Y	N	N	N	N	N	N	N	N	N	Y
<i>Data Locality Aware</i>	Y	Y	N	N	N	N	Y	N	Y	Y	N	Y	Y	Y
<i>Data splitting</i>	N	N	N	N	Y	N	N	N	N	N	N	N	N	Y
<i>Different dependencies</i>	N	N	N	N	N	N	N	N	N	N	N	N	N	Y
<i>SLA guarantee</i>	Y	Y	Y	Y	N	N	N	Y	N	N	Y	N	N	Y
<i>Profit optimization</i>	N	Y	N	N	N	N	N	N	N	N	N	N	N	Y

Y: covered; N: not covered

Li et al. [37] propose Big Provision, a provisioning framework, to effectively configure big data analytics systems. Chen et al. [32] address the challenges of delivering continuous Analytics-as-a-Service for realtime events. Wong et al. [46] propose Thrifty for parallel processing Database-as-a-Service to achieve a lower total cost with SLA guarantees on query results.

There are several widely used data analytics frameworks that support big data processing. The big data analytic frameworks that are studied in our work are Shark, Hive, Impala, and Tez/Stinger, which are built on top of the Hadoop Distributed File System [63]. Hadoop is an open-source software framework for reliable, scalable, and distributed computing. Shark is a new module in Spark, which allows interactive SQL on Hadoop systems [65]. Spark is an open-source fast computing framework to provide an easy-to-use programming model and leverages various applications [66]. Hive provides ad hoc query processing and summarizing of data on a data warehouse infrastructure [67]. Impala brings scalable parallel database technology to Hadoop, enabling multi-tenancy and low-latency SQL queries [68]. Tez offers fast processing

speed and data scalability to execute batch and interactive tasks [69]. Ousterhout et al. [142] systematically identify the performance bottlenecks of data analytics frameworks to quantify bottlenecks to enable researchers and practitioners make performance improvements.

There have been a number of related works for resource scheduling focus on different perspectives, scenarios, and techniques. Wang et al. [103] proposed a data-aware work stealing technique that is able to achieve efficient load balancing with data-aware scheduling. Xia et al. [79] studied a collaboration- and fairness-aware big data management problem to fairly place continuously generated data to distributed datacenters. Garg et al. [75] manage resources for heterogeneous workload in a datacenter and schedule resources for mixed workloads. Garcia et al. [41] introduce Cloudcompaas, an SLA-aware PaaS cloud platform to manage the complete resource lifecycle and offers a generic SLA model for cloud providers. However, this work does not consider big data analytics but general cloud computing applications. Among the related research works, Gu et al. [76] study the cost minimization problem for big data processing in distributed cloud datacenters. However, this work does not consider QoS requirements, SLA, or time optimization.

There are an extensive number of works focusing on delivering specific Analytics-as-a-Services, which enriches the BDAAs and services of AaaS platforms. Tordini et al. [56, 143] proposed data splitting techniques to process next generation sequencing datasets for complex workflows, which keep a high level of query accuracy. Zhang et al. [57] propose ND-based data splitting methods for large medical data processing in clouds. This approach splits large datasets to obtain smaller representative data for processing thereby offering higher flexibility, less resource costs, and faster response times.

4.6 Conclusion and Future Directions

Resource scheduling has significant meaning for AaaS platforms to maximize profits while delivering satisfactory AaaS to users with SLA guarantees. We model and formulate the data-splitting based resource scheduling problem. Moreover, we propose the admission control and profit optimization resource scheduling algorithms, which enable big data analytics with limited budgets and tight deadlines and provide profit optimized scheduling solutions with minimized query times and SLA guarantees.

Furthermore, we conduct extensive experiments to evaluate the performance of the proposed algorithms under different scheduling scenarios including realtime and periodic scheduling with varied SIs. Experimental evaluations show the admission control algorithm allows effective query admission while PO performs significantly better than HGS, AILP, and AGS in cost saving and profit enhancement in single schedule, for each BDAA, and for the entire query workload. The PO scheduling algorithm can better serve the profit maximization objectives while delivering AaaS services with SLA guarantees for query processing with full accuracy requirements for Scheduling Scenario 1. Moreover, we evaluate the performance of SPO with comparison to PO. The results show the SPO scheduling algorithm significantly increases the profits, reduces the resource costs, increases query admission rates, and decreases query processing times. SPO is a promising scheduling algorithm that leverages big data processing with limited budgets and tight deadlines and enables users to obtain time- and cost-effective analytics solutions for efficient decision making for Scheduling Scenario 2.

As part of the future work, we will continue working on 1) studying approximation query processing, 2) proposing effective data splitting and sampling techniques, 3) proposing SLA-based resource scheduling algorithms to allow query processing on large datasets with faster response times and reduced resource costs.

Acknowledgments

The authors wish to thank Dr. Bingsheng He and the reviewers of the 2016 IEEE International Conference on Big Data for their helpful feedback.

5 SLA-Aware and Data Sampling Based Profit Optimization of Resource Management

This chapter is originally derived from paper¹ that addresses Problem Scenario 3. In this chapter, we improve the work of Chapter 3 and Chapter 4 by enhancing AaaS solutions with further QoS guarantees based on accuracy requirements of query results for more reliable AaaS. This chapter makes the following key contributions. It provides efficient and effective multi-objective optimization admission control and resource scheduling algorithms that elastically and automatically provision large-scale cloud resources to schedule queries with SLA guarantees based on query deadline, budget and accuracy requirements.

The proposed algorithms offer profit optimization for AaaS providers and query time minimization for users. Data sampling-based scheduling methods are applied for processing samples of very large datasets and provide accuracy guarantees for analytical results based on the sampled data where accuracy can be traded-off for reduced costs and quicker responses when necessary. Experimental evaluations show that the proposed SLA and data sampling-based optimization resource scheduling algorithms significantly outperform the state-of-art scheduling algorithms and are able to admit more queries, produce higher profits, provision efficient resource configurations, generate less resource cost, offer increased flexibility, provide AaaS solutions with accuracy guarantees, and tackle big data analytics challenges under budget and deadline constraints. The chapter makes a significant contribution regarding the applicability, efficiency, flexibility, and performance improvement of multi-objective data sampling-based admission control and resource scheduling algorithms in supporting cost-efficient, fast and reliable AaaS delivery.

This chapter is originally derived from the following publication:

¹Y. Zhao, R.N. Calheiros, A.V. Vasilakos, J. Bailey, and R. O. Sinnott, “Profit maximization and time minimization admission control and resource scheduling for cloud-based big data analytics-as-a-service platforms,” in *Proceedings of 2019 International Conference of Web Services (ICWS)*, pp. 26-47, 2019.

Profit Maximization and Time Minimization Admission Control and Resource Scheduling for Cloud-Based Big Data Analytics-as-a-Service Platforms

Yali Zhao¹, Rodrigo N. Calheiros², Athanasios V. Vasilakos³, James Bailey¹, Richard O. Sinnott¹

¹The University of Melbourne, Australia

yalizhao.alice@gmail.com, baileyj@unimelb.edu.au, rsinnott@unimelb.edu.au

²Western Sydney University, Australia

r.calheiros@westernsydney.edu.au

³Lulea University of Technology, Sweden

athanasios.vasilakos@ltu.se

Abstract

Big data analytics typically requires large amounts of resources to process ever-increasing data volumes. This can be time consuming and result in considerable expenses. Analytics-as-a-Service (AaaS) platforms provide a way to tackle expensive resource costs and lengthy data processing times by leveraging automatic resource management with a pay-per-use service delivery model. This paper explores optimization of resource management algorithms for AaaS platforms to automatically and elastically provision cloud resources to execute queries with Service Level Agreement (SLA) guarantees. We present admission control and cloud resource scheduling algorithms that serve multiple objectives including profit maximization for AaaS platform providers and query time minimization for users. Moreover, to enable queries that require timely responses and/or have constrained budgets, we apply data sampling-based admission control and resource scheduling where accuracy can be traded-off for reduced costs and quicker responses when necessary. We conduct extensive experimental evaluations for the algorithm performances compared to state-of-the-art algorithms. Experiment results show that our proposed algorithms perform

significantly better in increasing query admission rates, consuming less resources and hence reducing costs, and ultimately provide a more flexible resource management solution for fast, cost-effective, and reliable big data processing.

Keywords: Optimization, Service Level Agreement, Analytics-as-a-Service, Admission Control, Resource Scheduling, Data Sampling, Big Data, Cloud Computing.

5.1 Introduction

Big data becomes an emerging trend that many organizations and companies are currently facing. A tremendous amount of data is being produced at an ever-accelerating rate from a myriad of sources such as IoT sensors, smart mobile devices, social media platforms amongst many other sources covering all aspects of society including health, commerce, government, and education. The need for advanced technologies to tackle data production, processing, and storage is clear. However, big data can only create values through the help of data analytics technologies to extract insights from such big datasets. It is fair to say that the success of many organizations, companies, and individuals lies heavily on big data analytics solutions.

Big data typically refers to massive volumes of structured, semi-structured, unstructured or real time data that exceed the processing and management capacities of traditional techniques [4]. Big data analytics is usually associated with cloud computing technologies. Cloud computing [144] provides a computing paradigm to dynamically provision resources for big data analytics solutions based on varying numbers of requests from users. As the data volumes increase to levels that exceed the storage and processing capabilities of individual computers, clouds allow to automatically and elastically provision cloud resources such as virtual machines (VMs) on-the-fly to process datasets for timely decision making as required by users and businesses. Cloud resources are typically provisioned in a pay-per-use model that enables users to only pay for the used resources. As such, big data analytics for small enterprises and individuals is possible, as they do not need to acquire expensive hardware and software directly or deal with the overheads with the management of such infrastructures.

AaaS platforms provide a way to tackle expensive resource costs and avoid long query times by leveraging automatic resource management capabilities. AaaS service delivery should ideally minimize the complexity of the resource management requirements for in depth knowledge of Big Data Analytics Applications (BDAAs) by users. It allows users from various domains to process big data with reduced times and cheaper costs based on SLA agreements between AaaS users and providers. In order to allow AaaS platforms to deliver SLA guaranteed AaaS for high user satisfactions, efficient and automatic resource scheduling is essential [11]. Resource scheduling is a core function of the AaaS platform and a central component to coordinate all the other AaaS platform components to deliver performance-oriented AaaS solutions. Our motivation in this paper is to provide optimal resource scheduling algorithms that can, on the one hand maximize the profits of AaaS providers, while on the other hand deliver AaaS to users within controllable budgets, factoring in deadline and accuracy guarantees for timely and reliable decision making.

A number of related research works [6, 11, 13, 56, 57, 74, 75, 76, 79, 80, 86, 90, 103, 109, 110, 140] have explored resource management in different perspectives and scenarios with support of various techniques in cloud environments. However, none of these works consider data sampling-based admission control and cloud resource scheduling algorithms to deliver AaaS with budget, accuracy, and deadline guarantees to support cost-efficient, reliable, and fast decision making. Resource scheduling of AaaS platforms faces several research challenges. Firstly, elastic and automatic resource provisioning is required to deal with dynamic online queries. Such requests can be stochastic in nature and have varying demands on the underlying cloud resources. Secondly, to support queries on very large datasets that cannot be processed under limited budgets or deadlines where accuracy can be traded-off for approximate processing [145], data sampling-based scheduling can be used. Thirdly, effective admission control should be applied to only admit queries satisfying Quality of Service (QoS) requirements that meet given SLAs. Finally, profit maximization and time minimization scheduling under various constraints represent a non-trivial multi-objective optimization problem, especially for complex BDAAs. This requires accurate modelling and formulation of the optimization problem.

To tackle the above challenges, the major **contribution** of our work is providing efficient and effective admission control and resource scheduling algorithms to

elastically and automatically provision cloud resources to schedule queries that serve the objectives of profit maximization for AaaS providers and query time minimizations for users while guaranteeing SLAs. The proposed algorithms apply data sampling-based admission and scheduling methods for big data processing under tight budget or deadline constraints. We formulate the resource management problem and implement the proposed admission and scheduling algorithms in the AaaS framework. To evaluate the performance of the proposed algorithms, we conduct extensive experiments. Experiment evaluations show that the proposed admission control and resource scheduling algorithms outperform the state-of-the-art algorithms in admitting more queries, creating higher profits, generating less resource costs with efficient resource configurations, and provide timely AaaS solutions with accuracy guarantees for reliable decision making under controllable budgets and tight deadlines.

5.2 Problem Statement

User submit queries $Query_s = \{Query_1, \dots, Query_M\}$ to a given AaaS platform. A query can be a scan query such as select patient name from patient table where patient age is greater than 30, or a more complex query such as a deep learning algorithm used to process large medical datasets to analyze the key factors that lead to diabetes. These queries request specific $BDAA_s = \{BDAA_1, \dots, BDAA_p\}$ that utilize a set of cloud resources $Resources = \{Resource_1, \dots, Resource_N\}$ and generating resource $Costs = \{Cost_1, \dots, Cost_N\}$. An example $BDAA$ can be a medicare application. A resource can be cloud containers, storage, single or clusters of VMs.

$Query = \{QoS, CR, BDAA, CH, AC, DE\}$. QoS requirements of a query request contain budget: the maximum costs to run a query, i.e., 1000 dollar; deadline: the latest time to deliver query result, i.e., 5 hours, accuracy: the confidence interval of the query result, i.e., 95% accuracy. CR is the cloud resources needed to run a query, i.e., EC2 VM instance or a cluster of VMs. $BDAA$ details a specific $BDAA$ requested for big data analytics, i.e., healthcare application. CH details the big data characteristics, such as the data distribution, data size, data type, accuracy requirement, and data locality. AC indicates whether the accuracy can be traded-off for reduced times and cheaper costs by applying effective data sampling methods. DE represents task dependency requirements including execution logic and sequence, i.e., father tasks and child tasks. Big data is

assumed to be pre-stored in cloud datacenters, where users pay data transfer and storage costs and hence the costs are not included in the cost models.

$BDAA = \{AT, P\}$. AT denotes the BDAA type and P is the BDAA profile. This includes mappings of the application cost, resource times required by queries including the processing and sampling time, and the resource configurations needed for queries. Obtaining application profiles of heterogeneous BDAAs typically requires expertise varied by different application domains. Therefore, reliable BDAA profiles are assumed to be maintained by third-party BDAA providers.

$Resource = \{RT, REC, NR, CC\}$. RT indicates the cloud resource type such as single or cluster of CPU optimized VMs. REC shows the cloud resource capacity including CPU, storage, memory. NR is the number of cloud resources needed to execute a query request. CC represents the cloud resource cost.

$Cost = \{RC, BC, PC, AC, PR\}$. RC is the overall cloud resource costs of the AaaS platform. BC represents the BDAA cost, which is assumed charged by BDAA providers as a constant value. PC represents the penalty cost that AaaS platforms need to pay users for SLA violation. AC represents the overall AaaS Cost (AC) that AaaS platforms charge users for utilizing AaaS, and PR represents the overall profits created by AaaS platforms. A fixed AaaS cost is assumed charged by AaaS platforms for a specific query based on its requirements of QoS, BDAA, and resource demands, and hence AC is a constant value.

The profit optimization problem to maximize the profits of the AaaS platform while minimizing the query times under various constraints is an NP-complete decision problem. The problem can be polynomially transformed to a mixed Integer Linear Programming (ILP) [120] problem with optimization formulation and modelling. We utilize the ILP modelling to formulate the multi-objective scheduling problem. AC and BC are constant values based on the cost models and PC is zero if AaaS is delivered with SLA guarantees. The profit of the AaaS platform is calculated by $PR = AC - RC - BC - PC$. Maximization of PR is transformed to minimize RC and deliver AaaS with minimized times subject to various constraints. We formulate the multi-objective Z based on a combination of individual optimization Objective X and Objective Y.

Objective X = $minimize(\sum_{j=1}^n C_j * t_j)$, where j denotes a resource; n represents a resource set; t_j is the number of resource required for j ; C_j is the unit cloud resource cost

of j . Objective X targets to minimize the RC , calculated as the product of the purchased time t_j and the unit cost C_j of all cloud resources.

Objective Y = $minimize (\sum_{i=1}^m s_i)$, where i is a query i ; m represents a set of queries, and s_i represent the query start time of i . Objective Y targets at finding an optimal solution of the resource scheduling to minimize the resource costs with optimized configuration to execute queries at its earliest times. This allows all queries starting earliest times to minimize t_j to save resource costs and improve the query performances with quickest responses.

Objective Z = $minimize (F_0 * \sum_{j \in n} (C_j * t_j) + \sum_{i \in m} s_i)$ is a combination of X and Y that aims to minimize RC while choosing a time-minimized scheduling plan, subject to optimization constraints (1)-(37). This combination contributes to a standard lexicographic problem [121]. The importance leading to the profit optimal scheduling solutions of individual objectives is $X > Y$. Coefficient $F_0 = \max(Y) - \min(Y) + 1$ is assigned to X to ensure the aggregated optimization with minimized individual objectives is consistent to the original optimization problem where changes of Objective X dominates all the changes in Objective Y .

Query Resource Capacity Constraints guarantee that the overall cloud resource times needed to process the complete datasets on resource j should be within the available time remained on resource j , (1), where x_{ij} is a variable with binary value that represents if query i is assigned to run on j ; R_{ij} is the cloud resource time needed by i to process the complete dataset using resource j ; t_j is the resource time of j that is to be purchased. For a given resource, T_j represents the remained resource time of j that is purchased in the previous schedule. For resources that is newly created, T_j is negative to deduct the resource creation time of j from t_j . When i executes on j , $x_{ij} = 1$; otherwise, $x_{ij} = 0$ shown in (2).

$$\sum_{i \in m, j \in n} (R_{ij} * x_{ij}) \leq t_j + T_j \quad (1)$$

$$x_{ij} = \begin{cases} 1, & i \text{ is assigned to execute on } j \\ 0, & \text{otherwise} \end{cases}, \forall i \in m; j \in n \quad (2)$$

Query Execution Sequence Constraints (3) ensure the unique query execution sequence as required by the optimal scheduling. b_{ik} is defined as a binary variable that defines the query execution sequence of i and k . If query i is scheduled to execute before k , $b_{ik} = 1$; otherwise, $b_{ik} = 0$.

$$b_{ik} + b_{ki} \leq 1, \forall i, k \in m \quad (3)$$

Query Dependency Constraints define query dependencies, $b_{ik} = 1$ when i is a dependent task requiring the execution results of k or a child task that can only start after k defined by (4). Query dependency constraints enable the definition of simple to complex task dependencies including bag of tasks, enterprise workflows, and scientific dataflows. For queries requiring data sampling for approximate processing, where data sampling and processing tasks are processed independently, query processing on sampled datasets can only start after data sampling finishes, as shown in (5).

$$b_{ki} = 1, \text{ if } i \text{ is a child/dependent task of } k, \forall i, k \in m \quad (4)$$

$$b_{ki} = 1, \text{ if } i \text{ processes data sample of } k, \forall i, k \in m \quad (5)$$

Query Deadline Constraints ensure scheduling solutions are generated before query deadlines for SLA guarantee purposes, as shown in Constraints (6) (7) (8) (9) (10) (11). Constraint (6) guarantee the unique query execution sequence of i and k that allows either b_{ik} or b_{ki} to be 1. Task i must execute either before or after task k when they are assigned to the same resource j as guaranteed by (7). Non-linear relationship (12) is transformed to linear relationship (7) through linearization, which guarantees if both i and k are scheduled to execute on j , $b_{ik} = 1, b_{ki} = 0$ when i is executed before k , or $b_{ik} = 0, b_{ki} = 1$ when i is executed after k . Big M method is utilized to derive Constraint (8) from non-linear constraint (13) to guarantee i should end before k starts if $b_{ik} = 1$. F_1 serves as a sufficient large constant that satisfies (14) [11]. Constraint (9) ensure query i finishes before its deadline D_i when executes on j at starting time s_i , derived from (15), which is non-linear constraint where F_2 serves as a sufficient large constant that satisfies (16). Non-linear relationship (17) is used to derive (10). F_3 serves as a sufficient large constant value that satisfies (18) to guarantee i finishes earlier than the purchased time of j to process the full dataset. The purchased time is accumulated from previous End of Purchased Time, EPT_j , and newly purchased time as t_j . Constraint (11) is generated based on the non-linear constraint (19). F_4 guarantees i starts at Earliest Available Time, EAT_j , of j , which satisfies (20) as a sufficient small constant. For an already created j with executing tasks, EAT_j represents the query finish time of j ; otherwise, EAT_j represents the current clock time. If j has not been created, EAT_j is the sum of creation time and clock time of resource j .

$$b_{ik} = \begin{cases} 1, & i \text{ is executed before } k \\ 0, & \text{otherwise} \end{cases}, \forall i, k \in m \quad (6)$$

$$b_{ik} + b_{ki} - x_{ij} - x_{kj} \geq -1, \forall i, k \in m; j \in n \quad (7)$$

$$s_i - s_k + F_1 * b_{ik} \leq F_1 - R_{ij}, \forall i, k \in m; j \in n \quad (8)$$

$$s_i + F_2 * x_{ij} \leq F_2 + D_i - R_{ij}, \forall i \in m, j \in n \quad (9)$$

$$s_i - t_j + F_3 * x_{ij} \leq F_3 - R_{ij} + EPT_j, \forall i \in m, j \in n \quad (10)$$

$$s_i + x_{ij} * F_4 \geq F_4 + EAT_j, \forall i \in m, j \in n \quad (11)$$

$$\begin{matrix} x_{ij} = 1 \\ x_{kj} = 1 \end{matrix} \left\{ \begin{matrix} \text{either} \\ \implies \end{matrix} \right\} \begin{matrix} b_{ik} = 1, b_{ki} = 0 \\ b_{ik} = 0, b_{ki} = 1 \end{matrix}, \forall i, k \in m, j \in n \quad (12)$$

$$b_{ik} = 1 \Rightarrow s_i + R_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (13)$$

$$F_1 \geq \max (s_i + R_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (14)$$

$$s_i * x_{ij} + R_{ij} \leq D_i, \forall i \in m, j \in n \quad (15)$$

$$F_2 \geq \max (s_i + R_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (16)$$

$$x_{ij} = 1 \Rightarrow s_i + R_{ij} \leq t_j + EPT_j, \forall i \in m, j \in n \quad (17)$$

$$F_3 \geq \max (s_i - t_j + R_{ij} - EPT_j) + 1, \forall i \in m, j \in n \quad (18)$$

$$x_{ij} = 1 \Rightarrow s_i \geq EAT_j, \forall i \in m, j \in n \quad (19)$$

$$F_4 \leq \min (s_i - EAT_j) - 1, \forall i \in m, j \in n \quad (20)$$

Data Sampling-based Query Budget Constraints guarantee that the resource cost to execute i on j is within the budget of B_i . If the entire dataset of query is to be processed in full, as shown in (21), C_{ij} represents the execution cost of query i on j that is within B_i . Otherwise, the accuracy of query may have to be sacrificed due to time and budget constraints, as shown in (22). SC_{ij} is the cost to sample and execute i on the data samples on j . This should be less than the budget of the query as B_i . Constraint (24) ensure if the cost of j exceeds the B_i j will be eliminated from the resource pool to enable the ILP solver to reduce the solution space and improve the algorithm performance.

$$C_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (21)$$

$$SC_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (22)$$

$$x_{ij} = 0, \text{ if } j \text{ cannot satisfy deadline of } i, \forall i \in m; j \in n \quad (23)$$

$$x_{ij} = 0, \text{ if } j \text{ cannot satisfy budget of } i, \forall i \in m; j \in n \quad (24)$$

Query Scheduling Times Constraints (25) guarantee the scheduling times of a query by specifying x_{ij} as 1 so that i is guaranteed to be scheduled to one cloud resource for execution in order to satisfy SLAs defined with users.

$$\sum_{i \in m, j \in n} x_{ij} = 1 \quad (25)$$

Data Locality Constraints ensure that queries can only execute on resources where datasets can be accessed. The aim is to avoid lengthy big data transfer times and

expensive big data transfer costs, as shown in (26). If a resource j has no access to the large datasets to be processed by i , $x_{ij} = 0$ guarantees i is not scheduled to j for SLA guarantees. While $x_{ij} = 1$ is set where the scheduling is required for the optimal scheduling solution.

$$x_{ij} = 0, \text{ if } j \text{ has no access to data of } i, \forall i \in m, j \in n \quad (26)$$

Data Sampling-based Query Resource Constraints, as shown in (27), ensure that the sum of data sampling time to obtain samples and the processing time to execute task on samples using j can be satisfied by available time of j . ST_{ij} represents the required Sampling Time (ST) to sample large dataset and process query i using j .

$$\sum_{i \in m, j \in n} (ST_{ij} * x_{ij}) \leq t_j + T_j \quad (27)$$

Query Accuracy Constraints determine the available resource configurations needed to execute queries without violating query deadlines and budgets. The accuracy of queries determines the data sample size and associated resources to execute the data sample. Increased accuracy requires that a larger data size will be selected and more computing resources will be consumed to process the larger sample. Resource j with a configuration not satisfying the accuracy requirements of queries will be eliminated from the selectable resource pool, as shown in Constraint (28), to enable the ILP solver to obtain the optimized solution in a reduced search space.

$$x_{ij} = 0, \text{ if } j \text{ cannot satisfy accuracy of } i, \forall i \in m; j \in n \quad (28)$$

Data Sampling-based Query Deadline Constraints (29) guarantee if $b_{ik} = 1$, i should finish sampling data and executing query on data sample no later than the start time of query k . Non-linear constraint (32) is used to generate (29) with F_5 satisfies (33) as a sufficient large constant. Constraint (30) ensure if i executes on j at s_i , they should finish sampling and execution no later than D_i . Non-linear constraint (34) is used to generate (30) with F_6 as a constant that is sufficient large satisfies (35). Non-linear relationship (36) is used to generate (31) with F_7 satisfies (37) as a sufficient large constant. This restricts if i executes on j , the data sampling and processing time of i should not exceeds the purchased resource time of j as the sum of EPT_j and t_j .

$$s_i - s_k + F_5 * b_{ik} \leq F_5 - ST_{ij}, \forall i, k \in m; j \in n \quad (29)$$

$$s_i + F_6 * x_{ij} \leq F_6 + D_i - ST_{ij}, \forall i \in m, j \in n \quad (30)$$

$$s_i - t_j + F_7 * x_{ij} \leq F_7 - ST_{ij} + EPT_j, \forall i \in m, j \in n \quad (31)$$

$$b_{ik} = 1 \Rightarrow s_i + ST_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (32)$$

$$F_5 \geq \max (s_i + ST_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (33)$$

$$s_i * x_{ij} + ST_{ij} \leq D_i, \forall i \in m, j \in n \quad (34)$$

$$F_6 \geq \max (s_i + ST_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (35)$$

$$x_{ij} = 1 \Rightarrow s_i + ST_{ij} \leq t_j + EPT_j, \forall i \in m, j \in n \quad (36)$$

$$F_7 \geq \max (s_i - t_j + ST_{ij} - EPT_j) + 1, \forall i \in m, j \in n \quad (37)$$

5.3 Admission Control and Resource Scheduling

Big data analytics faces the challenges of tight budgets and/or deadlines, where queries cannot always be fully admitted whilst satisfying all QoS requirements with SLA guarantees during AaaS delivery. To tackle such research challenges, data sampling-based optimization algorithms are proposed that process data samples and return AaaS solutions in a faster manner with significantly reduced resource costs and enhanced profits.

5.3.1 Admission Control

Queries are submitted online to the AaaS platform from various domain users. The admission controller iteratively admits each query. The pseudo code of the admission control algorithm is shown in Algorithm 1. The admission controller first checks if BDAA is available and the associated dataset is accessible. If so, it further estimates if the QoS requirements of budget, deadline, and accuracy can be satisfied by any existing cloud resource configuration in the AaaS platform or whether it can be brokered from third party resource providers. It searches all resource configurations in the registry of cloud resources. The admission controller calculates the Estimated Cost (EC) and Estimated Time (ET) for each configuration to execute a query satisfying the given QoS requirements. If such configuration is found, the admission controller admits the query as *fullQuery* and adds the query to the admission queue. (Lines 1-11).

If the admission controller cannot admit a query with tight budgets and/or deadlines, approximate processing of the query is considered only if the results are meaningful without affecting reliable decision making. The admission controller estimates if QoS requirements can be satisfied through the support of data sampling considering the overall query cost that is calculated as the sum of the Sampling Cost (SC) and the Data Processing Cost (DPC). The overall query time is calculated as the sum of data Sampling Time (ST) and Processing Times (PT). Furthermore, the admission controller

estimates whether the query Accuracy (A) can be fulfilled by processing the data samples on given resource configuration. (Lines 12-19).

BDAA profiles provisioned by third-party providers contain information of the BDAA costs, the required resource times including sampling times and query times, as well as the resource configurations for different query requests. BDAA application profiles are the bases for the AaaS platform to make accurate estimation of query processing time, accuracy, and cost for admission and scheduling decisions. BDAA costs for the same BDAA can be different for different versions, e.g. sequential processing versions usually costs less than parallel processing versions.

Based on the application profiles, we can obtain the query processing times for given resource configurations, the accuracy on given resource configurations, and the BDAA costs to decide whether QoS requirements of queries can be fulfilled. This is achieved by using estimation method in the following way: for a given query, based on the accuracy requirement, profiles satisfying accuracy can be preliminarily selected. After considering budget requirements, all of the profiles that are still possible to be selected to execute a given query within user specified budgets are selected. The admission controller further calculates the overall costs including BDAA costs and resource costs for available BDAA profiles to satisfy the budget constraints of queries.

For a given query that supports approximate processing, if all QoS requirements can be satisfied by at least one cloud resource configuration, such a query is acceptable and executable as apprQuery and the SLAs are established by the SLA manager; otherwise, the query has to be rejected to avoid significant penalty costs caused by SLA violations. Afterwards, the AaaS platform utilizes optimization algorithm to make scheduling decisions to maximize the profits while minimizing query response times in the AaaS platform. After the data sampling-based admission control, rejection reasons are given to users to subsequently modify query specifications for potential resubmission. (Lines 20-31).

Algorithm 1: Query Admission Control Algorithm

Input: big data analytics requests/ queries, BDAAs, cloud resources, resource configurations

Output: query acceptance or rejection decisions

- 1: **for** every submitted query $\in$ submitted big data analytics requests
- 2: requested BDAA by the query $\leftarrow$ broker/search the BDAA registry

```
3:  if the BDAA is brokerable/available && big data can be accessed
4:    configurations  $\leftarrow$  search the resource registry for all configurations
5:    if configurations >0
6:      for every configuration  $\in$  all available resource configurations
7:        ET  $\leftarrow$  estimate the execution time on the resource configuration
8:        EC  $\leftarrow$  estimate the query cost on the resource configuration
9:        if EC < budget && ET < deadline
10:         admQueries  $\leftarrow$  query admission as fullQuery
11:        end if
12:        else if the query request allows sacrificing accuracy for reduced times and
costs
13:         apprQueries  $\leftarrow$  get queries support approximate processing
14:         for every query request in the apprQueries
15:           SC  $\leftarrow$  estimate the cloud resource cost of data sampling
16:           ST  $\leftarrow$  estimate the required time of data sampling
17:           DPC  $\leftarrow$  estimate the cloud resource cost to process data samples
18:           PT  $\leftarrow$  estimate the time to process data samples
19:           QA  $\leftarrow$  estimate data accuracy to process data samples
20:           if SC +DPC < budget & ST +PT < deadline & A > accuracy
21:             admQueries  $\leftarrow$  query admission as apprQuery
22:             avaiResources  $\leftarrow$  update the available cloud resources
23:           end if
24:         end for
25:       end if
26:     end for
27:     if the query request is not successfully admitted
28:       reject the data analytics request with rejection details
29:     end if
30:   end if
31:end for
```

Admitted queries can be assigned to existing resources for execution if the current resources have sufficient capacity; otherwise, new cloud resources are created to execute the query following the SLA agreements. The optimal query assignment and resource provision decisions are ultimately provided by the resource scheduler for profit maximization and query time minimization.

5.3.2 Resource Scheduling

To provide timely, reliable, and cost-effective scheduling solutions for the AaaS platform, a data SAMpling-based Profit Optimization (SAPO) scheduling algorithm is

put forward. SAPO offers scheduling with SLA aware and data sampling-based methods to provision resources to query processing on data samples to meet QoS requirements of budget, deadline, and accuracy. The pseudo code of the SAPO resource scheduling algorithm is shown in Algorithm 2.

SAPO allows data analytics results to be returned in a time-efficient and reliable manner with controllable resource costs that can benefit users with limited budgets and with timely decision-making requirements where accuracy bounds are necessary for reliable decision making. Resource scheduling is the core function of the AaaS platform required to coordinate all the other components to deliver satisfactory services to users through the SAPO algorithm.

The AaaS service delivery scenario starts when users submit queries to the AaaS platform. Each query requests for specific BDAA to analyze the data. For a BDAA, the SAPO scheduler first obtains information from all queries and resources supporting the BDAA, along with the up-to-date information on the resource configurations from the AaaS platform as the input to the scheduler.

The SAPO scheduler calls the admission controller to admit query requests. Queries are admitted by the admission controller in the following two scenarios. Scenario 1: sufficient budgets as well as deadlines are given by users to execute the queries that can meet SLA guarantees on AaaS delivery based on processing the full datasets, named as fullQueries. Scenario 2: tight deadlines and/or limited budgets are given by users. In this way, processing the entire dataset is not permitted. If accuracy of queries can be traded-off where effective approximate processing is supported for reliable data analytics results, data sampling is used to approximately process smaller sampled datasets to tackle the time and cost challenges with reliable accuracy bounds presented to users, such queries are named as apprQueries. SAPO schedules queries by applying the SAPOOptimization method that adopts and implements the formulation of the optimization scheduling problem by applying the ILP programming model with Objective Z subjecting to a range of optimization scheduling constraints (1) - (37).

Algorithm 2: SAPO Cloud Resource Scheduling Algorithm

Input: user submitted big data analytics requests, cloud configurations, all available BDAAs, cloud resources.

Output: optimized | heuristic resource scheduling solution.

```
1: for each requested BDAA  $\in$  all available BDAA's
2:   admQueries  $\leftarrow$  admissionControl (queries, resourceConfigurations, BDAA)
3:   fullProcessingTasks  $\leftarrow$  obtain all fullQueries from admitted queries
4:   apprQueries  $\leftarrow$  obtain all approximate queries from admitted queries
5:   for every query request in apprQueries
6:     dataSamplingTasks  $\leftarrow$  create data sampling tasks
7:   end for
8:   anaQueries  $\leftarrow$  fullProcessingTasks + dataSamplingTasks
9:   if (anaQueries > 0)
10:    resources  $\leftarrow$  obtain all existing cloud resources running BDAA
11:    heuCloudResources  $\leftarrow$  selectSampledResourceHeuristic (resources,
    configurations, anaQueries, BDAA)
12:    optimalSolution  $\leftarrow$  SAPOOptimization (anaQueries, BDAA's,
    heuCloudResources)
13:    if a feasible solution is obtained before timeout setting
14:      apply the optimal solution to provision resources for query execution
15:    end if
16:    else
17:      heuristicSolution  $\leftarrow$  DTheuristic (heuCloudResources, anaQueries,
    BDAA)
18:      apply the heuristic solution to run queries
19:    end else
20:    activeResources  $\leftarrow$  update existing active resources running BDAA
21:    scaleDown (activeResources)
22:  end if
23: end for
24: Procedure: SAPOOptimization (anaQueries, heuCloudResources, BDAA)
25:   getUpdateToDateInfo (anaQueries, heuCloudResources, BDAA)
26:   defineOptimizationScheduleObjectives (Objective Z)
27:   defineOptimizationScheduleConstraints (Constraints (1)-(37))
28:   defineOptimizationTimeOut (schedulingTimeout)
29:   solveOptimizationScheduleProblem ()
30:   optimalSolution  $\leftarrow$  getOptimizationScheduleSolution ()
31: Procedure: DTheuristic (admQueries, heuCloudResources, BDAA)
32:   sort admQueries based on maximum delay time
33:   schedule admQueries to resources with minimized earliest start time
34: Procedure: selectSampledResHeuristic (resources, BDAA's, configurations,
    admQueries)
35:   avaiCloudResources  $\leftarrow$  obtain all available cloud resources
36:   exeCloudResources  $\leftarrow$  obtain resources from avaiCloudResources
37:   newCloudResources  $\leftarrow$  generate new resources to execute queries
38:   heuCloudResources  $\leftarrow$  exeCloudResources + newCloudResources
```

For queries that cannot be executed in full by processing the entire datasets under tight budgets and/or deadlines, the admission controller attempts to admit *apprQueries*. If the sampling technique can deliver satisfactory AaaS services, such *apprQueries* are admitted to the AaaS platform. The SAPO scheduler then generates *anaQueries* from all admitted analytics tasks. The scheduler then provisions *heuCloudResources* applying the *selectSampledResHeuristic* method. Furthermore, the SAPO scheduler utilizes the proposed *SAPOOptimization* algorithm by applying the ILP programming model to create profit-maximization and cost-minimization scheduling solutions. (Lines 1-12).

If the generated optimization solutions are feasible that is returned before system-defined scheduling timeout, the optimal solution is then utilized to guide query execution and resource provisioning in the AaaS platform. (Lines 13-16). If no scheduling solution is generated before the timeout setting of the optimization algorithm, a heuristic approach named as *DTHuristic* is used to generate alternative heuristic solutions to avoid SLA violations caused by the failure of query execution. After the scheduling solution is created for the current schedule, the SAPO scheduler triggers auto-scaling to downgrade the capacity by terminating active cloud resources that are idle to save costs at checkpoints by the *scaleDown* method. (Lines 17-23).

The ***SAPOOptimization*** method (Lines 24-30) first obtains the current platform information regarding the queries, resources, and BDAAAs that are used as the input to the ILP solver. SAPO defines and implements the optimization objectives and constraints. SAPO enables the objective function to maximize the profits and minimize query responses for the AaaS platform. SAPO is subject to various constraints (1)-(37) based on the mixed ILP formulation of the optimization problem. SAPO support data sampling and SLA aware resource scheduling solutions that are designed to tackle the resource scheduling challenges for fast, reliable, and cost-effective optimization solutions to improve the performance and quality of service delivery by the AaaS platform.

The ***DTHuristic*** method (Lines 31-33) applies a maximum delay time-based heuristic algorithm to map *admQueries* to *exeCloudResources* and executes queries at the earliest available start time on the selected cloud resources.

The ***selectSampledResourceHeuristic*** method (Lines 34-38) significantly reduces the Algorithm Running Time (ART) of *SAPOOptimization* by reducing the problem

search space to efficiently generate data sampling-based optimized resource scheduling solutions. The method first selects *avaiCloudResources* as the available cloud resources selected from the existing cloud resources. It then selects cloud resources with the capacity to execute at least a *fullQuery* or an *apprQuery* satisfying its QoS requirements utilizing *avaiCloudResources* that is defined as *exeCloudResources*. The *selectResourceHeuristic* further generates new resources to execute the submitted queries if the platform does not have sufficient resource capacity to execute newly submitted query requests. The method finally generates *heuCloudResources* as the summation of *exeCloudResources* and *newCloudResources* that are then input to the *SAPOOptimization* method. The *heuCloudResources* is able to provision sufficient resources that enable *SAPOOptimization* to find feasible solutions to schedule queries satisfying SLAs. The *heuCloudResources* method provisions sufficient resources with capacity close to the optimal configuration created by *SAPOOptimization*. The aim is to reduce the search space to enable *SAPOOptimization* to return optimal solutions in a timely, reliable, cost-effective way.

We compare the performance of SAPO to a Profit Optimization (PO) scheduling algorithm. PO [11] serves as a suitable comparison algorithm as it serves the same objective as SAPO to maximize the profits for the AaaS platform while providing AaaS with minimized queries times for users. Moreover, PO also builds on ILP-based formulation and provides optimal resource scheduling solutions in the AaaS platform. PO applies SLA-based scheduling mechanisms that are able to deliver optimal solutions subject to different constraints, which are: resource capacities, budget requirements, deadline requirements, task execution times and sequences, task dependencies, and data locality constraints [11]. Since PO is not able to support sampling-based scheduling solutions to process big data under constraints of tight deadlines and budgets, it has the limitations in only admitting and scheduling queries with sufficient budgets and deadlines. Thus, PO is not able to tackle big data challenges incurring expensive costs or lengthy processing times where fast, cost-effective, and reliable AaaS is required by decision making in BDAA domains such as banking and stock market. To face such challenges, SAPO serves as the ideal optimization resource scheduling algorithm that is able to deliver optimal solutions not only support big data analytics scenarios enabled by PO but also support big data analytics within tight deadline and limited budget

constraints by applying data sampling-based scheduling for timely, cost-efficient, and reliable AaaS solutions in the cloud computing environments.

5.4 Performance Evaluation

We conducted experimental evaluations to analyze the efficiency of the profit maximization and cost minimization algorithms including SLA guarantees, query admission control, cost saving, resource configuration, profit enhancement, accuracy analysis, and ART analysis.

Experiment Setup: We built the AaaS framework using CloudSim [122] and utilize IBM CPLEX 5.5 as the optimization ILP solver [134]. We conducted experiments for real time and periodic cloud resource scheduling with different Scheduling Intervals (SIs).

Resource Configuration: 4 datacenters are simulated. Each datacenter consists of 500 nodes while each node contains 400 CPU, 10PB storage, 30TB memory, and bandwidth of 10GB/s. Six types of VMs are considered as memory optimized Amazon EC2 VMs: r4.large, r4.xlarge, r4.2xlarge, r4.4xlarge, r4.8xlarge, and r4.16xlarge [123]. A resource can be a CPU core, a single or a cluster of VMs. The unit for memory, storage, cost, and SI is GiB, GB, dollar, minute accordingly.

Data Analytics Workload utilizes Big Data Benchmark [124] and BlinkDB data sampling workload [140]. The big data benchmark provides query response times and the cloud resource configurations using Amazon EC2 VMs to execute on big data analytic frameworks with detailed data size, data location, and data type of datasets. The BlinkDB technique samples the original datasets for sampling processing of the data. The BlinkDB workload provides the resource times, configurations to sample big data with query times and error bounds on accuracies. Based on the big data benchmark and BlinkDB, the times for data sampling and query processing on various configurations using BDAAAs are modelled.

Query Information: *Submission time* is generated using one-minute mean Poisson arrival interval. *Query type* has 4 types including scan, join, aggregation, and user defined function. BDAAAs has 3 types including Hive on Hadoop as BDAA 1, Hive on Spark without caching as BDAA 2, and Hive on Spark with caching as BDAA 3. *Resource time* contains the resource requirements of queries. Two types of deadline and budgets are considered: tight, which are generated with a Normal Distribution of (3,

1.4), and loose, which are generated with a Normal Distribution of (8, 3). *Accuracy* is generated-based on the BlinkDB sampling workload which details the guaranteed accuracy with given response times and cloud resource costs to execute the approximate queries with error bounds of results. 5 types of accuracy are considered that are 100%, 99%, 95%, 90%, and 85%.

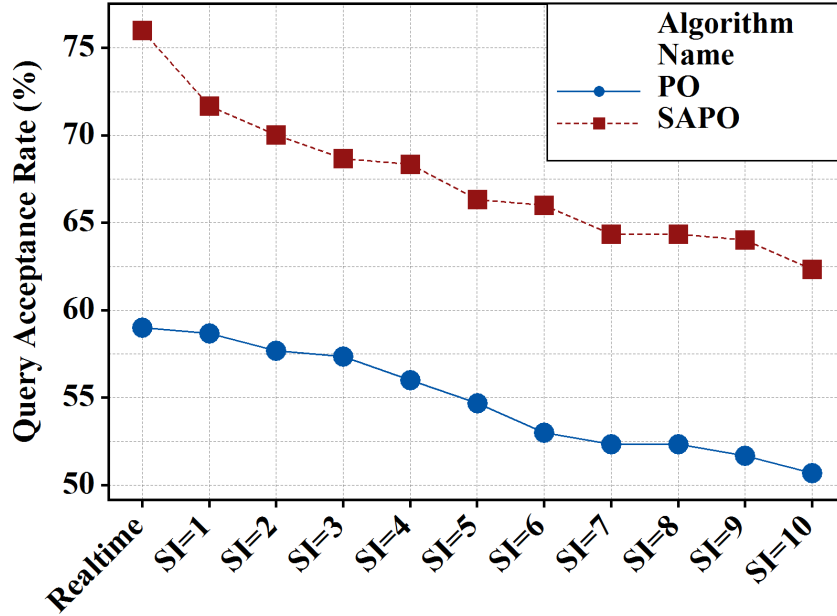


Figure 5.1: Admission Rates of SAPO and PO.

1. Admission Control and SLA Guarantees: To evaluate the algorithm performance of efficient and effective admission control, we conduct experiments for real time and periodic scheduling where SI is in the range of [1, 10]. We compare the algorithm performance of SAPO compared to PO with results shown in Figure 5.1. We can see that the SAPO is able to admit more queries for processing for both real time and periodic resource scheduling with an increased query admission rate in the interval of [12%, 17%]. Higher admission rate for processing queries can create higher profits, increase user satisfactory levels, and enlarge markets by processing more data analytics requests, and hence is highly preferred. Moreover, results also show that all admitted queries by SAPO and PO are processed with SLA guarantees, which proves the effectiveness of the admission control.

2. Profit Enhancement: We evaluate the profit enhancement advantages of SAPO for real time and periodic scheduling scenarios, as shown in Figure 5.2. Results show

that SAPO creates significantly higher profit than PO for all scheduling scenarios. The increased profit interval is [27.5%, 38.6%]. The enhanced profits come from higher query admission rates and reduced resource consumption from data sampling-based query processing. The profits show a decreasing trend for both SAPO and PO due to the rejection of queries with tight deadlines for periodic SIs. Real time scheduling is not realistic for all online resources since it would generate unnecessarily large computing usage and hence periodic scheduling approaches are supported to schedule queries that arrive during specified SIs. We find SI=1 is the most suitable periodic SI that can, on the one hand admit more queries, while on the other hand it reduces frequent scheduling computations, which is important for higher admission rates and higher profit enhancements. We further analyze the data processing methods of SAPO and PO and provide details of the data processing methods. We find that the SAPO algorithm supports both full data processing and sampling-based processing of the datasets while the PO scheduling algorithm only supports full data processing. The percentage of queries that are processed using data sampling methods for the SAPO algorithms are in the interval of [16.6%, 22.4%], which shows an increased query rate that benefits creating higher profits for SAPO.

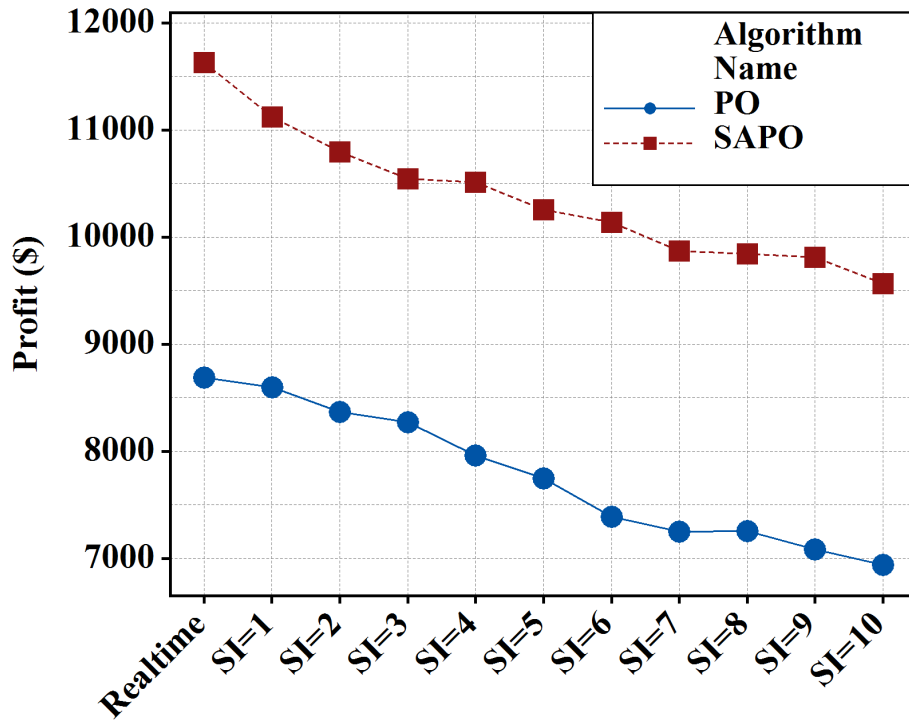


Figure 5.2: Profits of SAPO and PO.

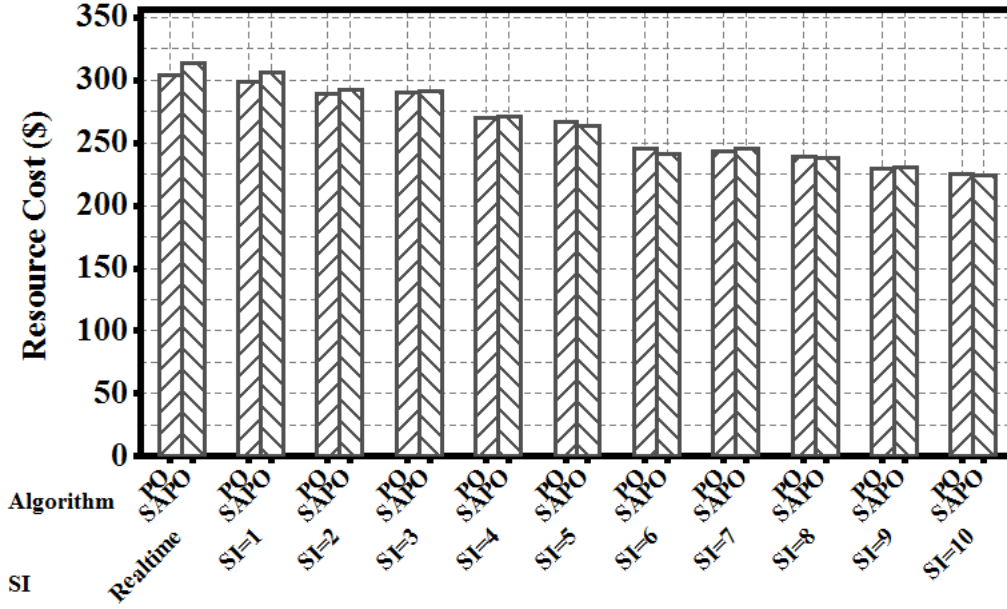


Figure 5.3: Resource Costs of SAPO and PO.

3. Cost Savings: Resource cost is another key performance indicator for algorithm performance. As shown in Figure 5.3, we can see that the resource costs of both SAPO and PO are similar. The increased cost rates of SAPO compared to PO are in the interval of $[-0.02\%, -0.3\%]$. Under the condition that SAPO admits $[12\%, 17\%]$ more queries and creates higher profits of $[27.5\%, 38.6\%]$, such similar resource consumptions and costs clearly indicate the performance advantages of SAPO in cost saving. SAPO is able to reduce costs and increase profits through better optimal decision making. SAPO allows to better utilize resources on larger number of queries with a more comprehensive optimization solution.

4. Resource Configuration: The resource configurations of SAPO and PO for real time and periodic scheduling are shown in Figure 5.4. There is a trend whereby r4.large and r4.16xlarge are more frequently utilized by both SAPO and PO. This is consistent with the query workload property whereby the workload is generated with query deadlines and budgets in two categories: tight and loose. For tight deadlines, r4.16xlarge can offer higher processing capacity to meet deadline requirements while r4.large can reduce unit resource waste for tight budgets when VMs are idle, hence both r4.large and r4.16xlarge are utilized more frequently. We can also see that the SAPO algorithm has an overall higher number of VMs utilization due to the higher number of admitted queries that require more cloud resources to provide the required computing

power for the query workload. We notice the decrease in resource consumption trend for both PO and SAPO due to the decreased query admission rates and hence decreased resource requirements. Decreased resource consumption also comes from the higher number of queries to be scheduled in increased SIs with resultant improvements in resource utilizations.

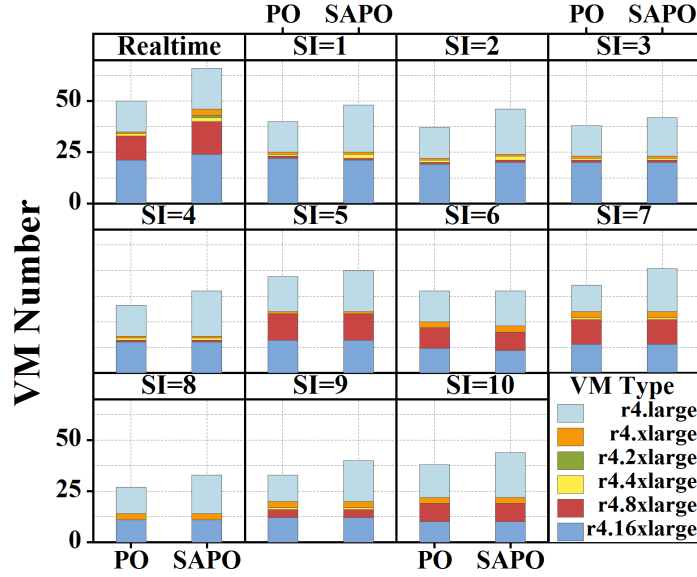


Figure 5.4: Resource Configurations of SAPO and PO.

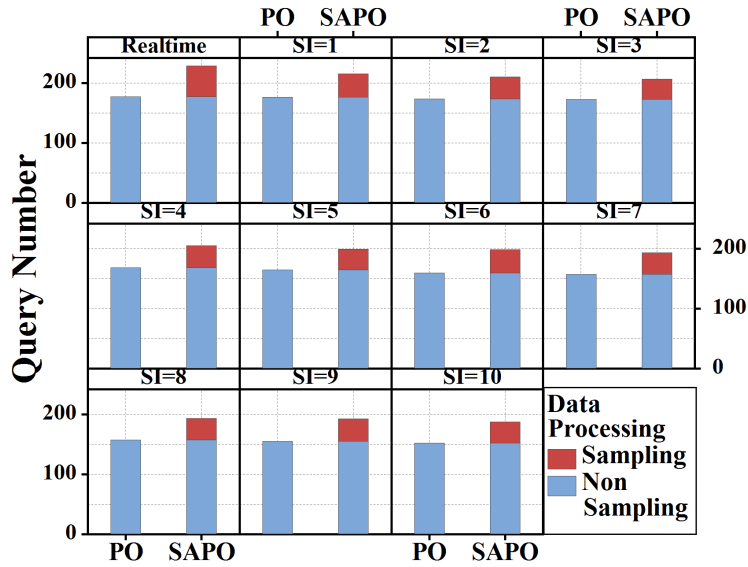


Figure 5.5: Query Number of Data Sampling and Non Sampling Processing of SAPO and PO.

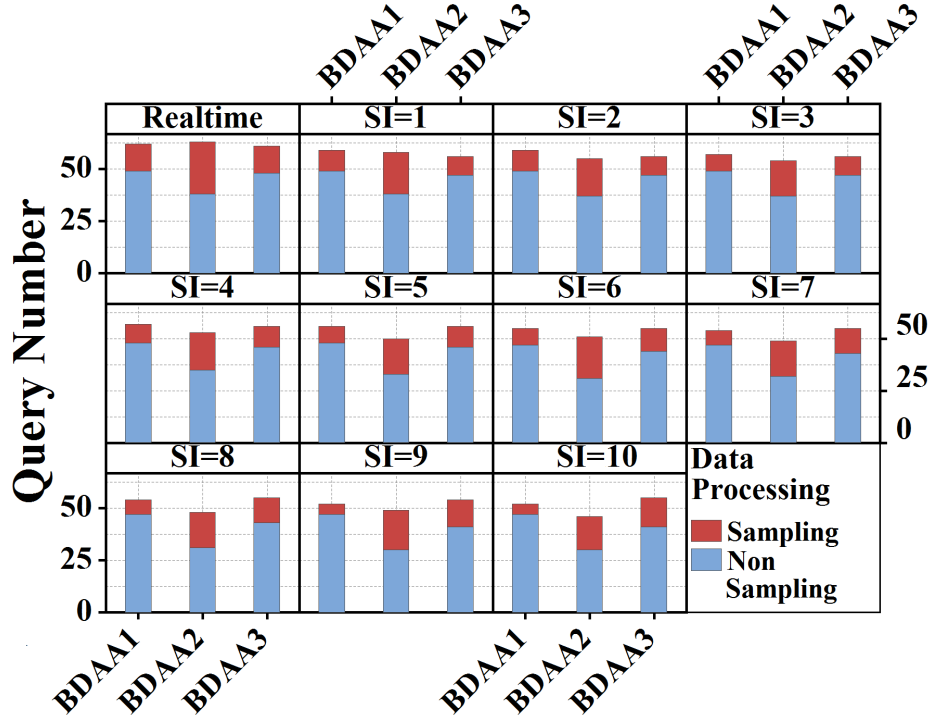


Figure 5.6: Query Number of Sampling and Non Sampling Processing for Each BDAA.

5. Data Processing Method Analysis: We further analyze the data processing methods of SAPO and PO for better understanding of the performance advantages of SAPO. We obtain the data processing methods of all queries for both SAPO and PO. Results show that the SAPO algorithm supports both full data processing and data sampling-based processing of the datasets while the PO scheduling algorithm only supports full data processing for all BDAAAs, as shown in Figure 5.5. The percentage of queries that are processed using data sampling methods for the SAPO algorithms are in the interval of $[16.6\%, 22.4\%]$, which shows that increased query rates create higher profits for SAPO.

After the overall analysis of the algorithm performance for all BDAAAs, we further analyze the algorithm performance of SAPO for each BDAA. The number of queries processed by different BDAAAs for SAPO and the details of the data processing method are shown in Figure 5.6. The number of queries with approximate processing is shown in Figure 5.7. We can see that although admitted queries that support approximate processing are in the interval $[80.4\%, 81.2\%]$ in Figure 5.8, SAPO only applies

sampling-based scheduling on queries when the QoS requirements cannot be fulfilled to prioritize highly accurate query processing needed for reliable decision making. As a result, approximately processed queries are in the interval [16.5%, 22.4%]. SAPO prioritizes the processing of the entire datasets to provide full accuracy of data analytics results to help users make better strategies and decisions while it deprioritizes the profit gaining created by approximate processing. SAPO aims to help the AaaS platform to achieve the best performance for higher user satisfaction and enlarge market share by higher quality AaaS delivery.

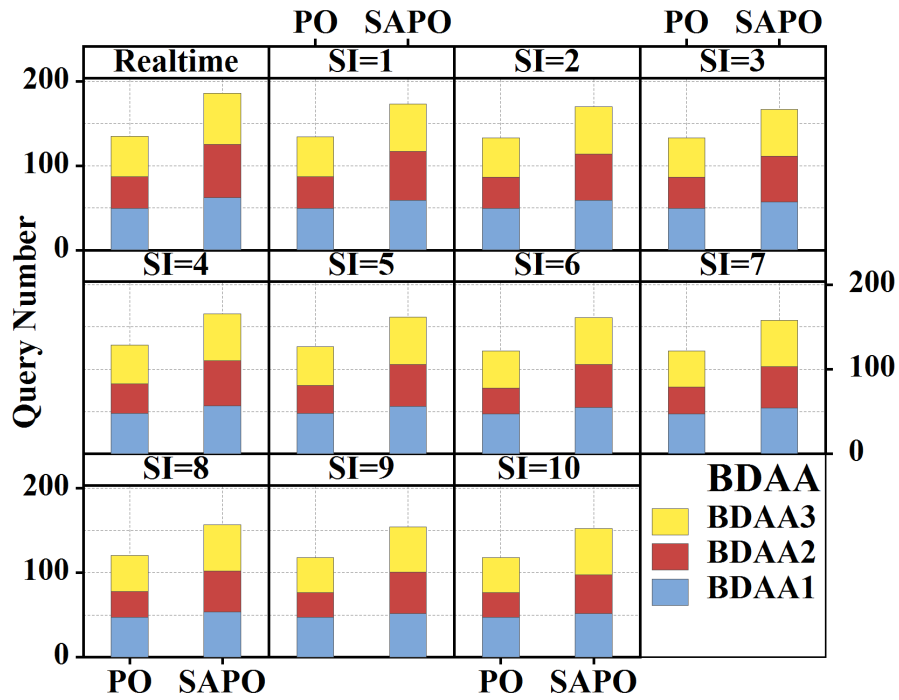


Figure 5.7: Approximate Query Processing Number for Each BDAA.

6. BDAA Analysis: After the algorithm performance analysis for the entire query workload processed by all BDAA, we further detail the performance evaluation results for each BDAA to show the performance advantages of SAPO. The results show that SAPO outperforms PO for each BDAA with real time scheduling and periodic scheduling using real time scheduling and SI=1 as examples. The results also show that SAPO is able to admit more queries for both real time and SI=1 with an increased query acceptance rate in the interval of [15.8%, 42.4%] for BDAA 1-3. The profits created by the SAPO algorithm for BDAA 1-3 are [22.4%, 41.7%] significantly higher than PO as

shown in Figure 5.9. Moreover, we further analyze the resource costs for SAPO in Figure 5.10. With all above performance advantage for query admission and profit enhancement, the resource cost of SAPO is only [0.01%, 0.03%] higher than the PO algorithms for BDAA 1-3, resulting in a significant performance advantage and cost saving. The results also show that SAPO is able to significantly outperform PO for the entire query workload with higher admission rates, higher profits, and lower resource costs. Experimental evaluations also show SAPO significantly outperforms PO for each BDAA for both real time and periodic scheduling.

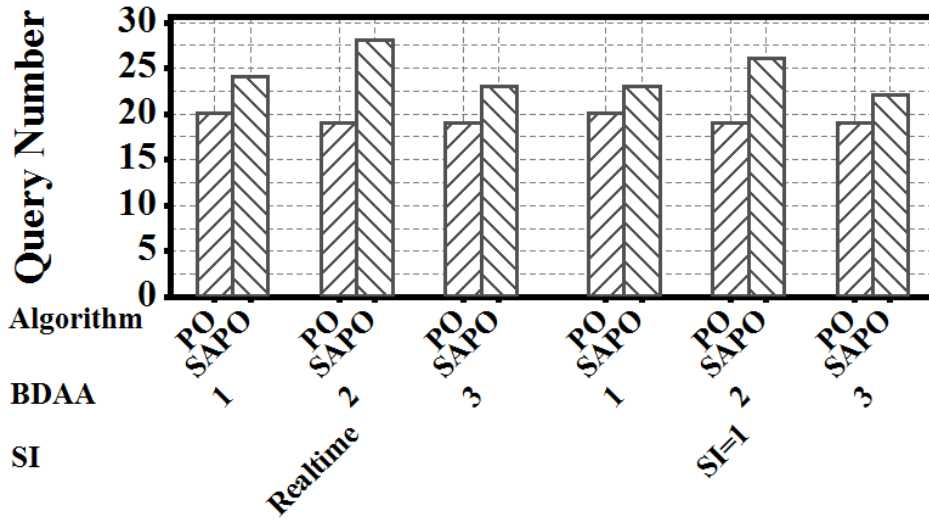


Figure 5.8: Query Acceptance Number for BDAAAs.

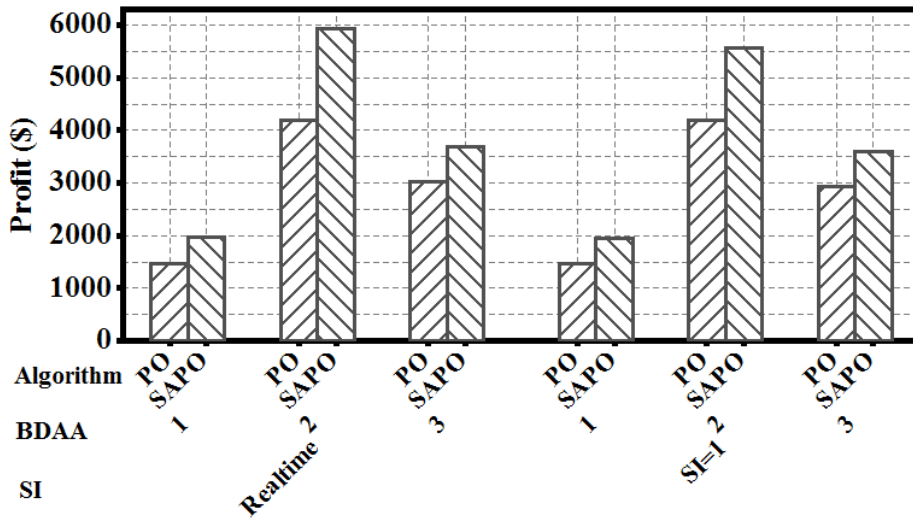


Figure 5.9: Profits of SAPO and PO for Different BDAAAs.

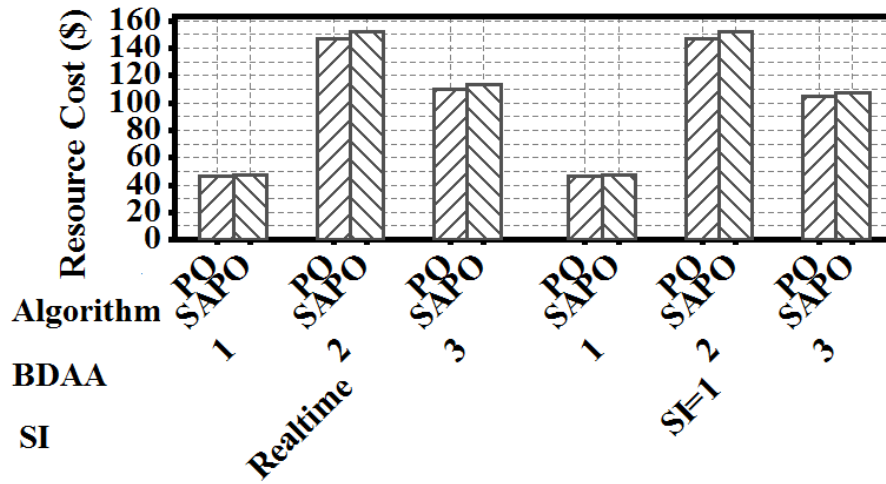


Figure 5.10: Resource Costs of BDAAs of SAPO and PO for Realtime and SI=1.

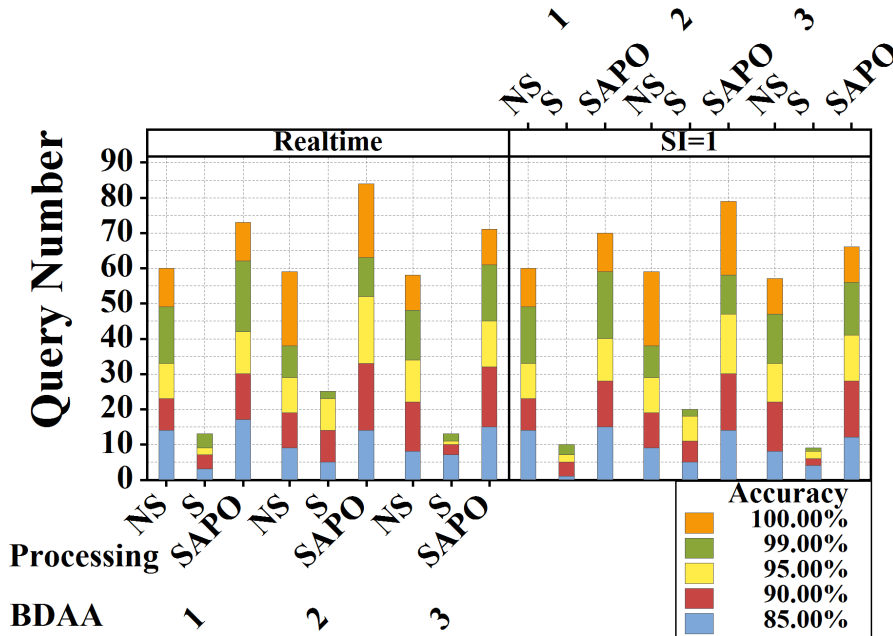


Figure 5.11: Accuracy Analysis.

7. Accuracy Analysis: We further analyze the accuracy of queries for different query processing methods with results shown in Figure 5.11. We can see that for data Sampling (S) based processing, Non-Sampling (NS) based processing, and SAPO processing queries with both S and NS-based methods, the number of queries processed with different accuracy requirements varies for different BDAA. All of the accuracy requirements of the queries are fulfilled with 100% SLA guarantees for both SAPO and

PO algorithms. For NS-based processing, we notice that though the query percentage with full accuracy requirements is in the interval of [17%, 36%], NS-based processing method of SAPO does not support approximate processing but only supports full processing of queries. Furthermore, for SAPO, despite [75%, 86%] queries supporting approximate processing, SAPO only process [14%, 30%] of queries with the data sampling-based processing method. The reason is that SAPO prioritizes fully accurate query processing for better decision making though approximate query processing thereby creating higher profits and reducing resource costs. SAPO aims to help the AaaS platform to improve the quality of AaaS delivery for efficient and reliable problem solving and decision making for users in various application domains.

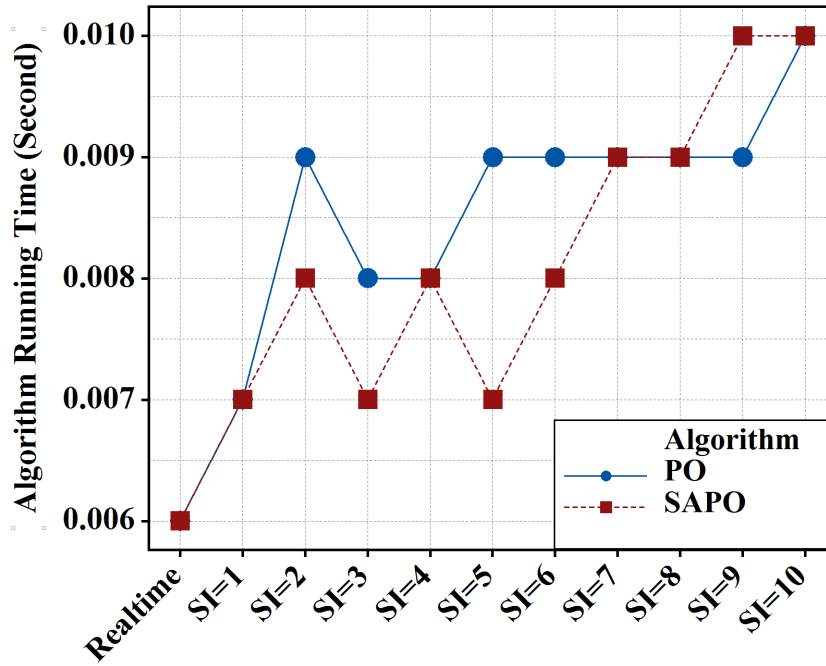


Figure 5.12: ART Analysis.

8. **ART Analysis:** The average ART of SAPO and PO algorithms for real time and periodic scheduling are shown in Figure 5.12. The results show an overall increasing trend for ART as more queries are scheduled and more optimization constraints need to be applied for more complex optimization problems with larger inputs and hence a larger solution space. Both SAPO and PO can deliver resource scheduling solutions within 6-10 milliseconds that illustrates the efficiency of the scheduling algorithms. As such, ART does not limit the SAPO scheduling algorithm in delivering effective and

efficient resource scheduling solutions. In order to prevent an unexpected large number of queries arriving so that the solution space and input scale exceed the capacity of the optimization algorithms to deliver solutions within SIs, heuristic algorithms are applied as backup solutions. These are able to make near optimal scheduling solutions to guarantee SLAs for the AaaS delivery. The backup heuristics enable the AaaS platform to provide comprehensive scheduling solutions for unexpectedly large query arrivals to mitigate risks and support AaaS delivery with SLA guarantees.

5.5 Related Work

We focus on providing optimization algorithms to support AaaS platforms to deliver cost-effective, timely, and reliable AaaS for better problem solving in various application domains.

Zhao et al. [6] study an SLA-based admission control and resource scheduling algorithm that offers a cost-effective solution to increase profits for AaaS providers. However, the solution can be sub-optimal due to two-phase resource scheduling. Zhao et al. [11] propose a profit optimization cloud resource scheduling algorithm that is able to maximize profits for AaaS providers with minimized query processing times with SLA guarantees for users, however, this solution cannot tackle big data challenges with limited times and costs. Zhao et al. [13] addresses the big data analytics challenge of resource scheduling under limited deadlines and budgets where accuracy can be traded-off by applying data splitting-based query admission control and profit optimization cloud resource scheduling algorithms. The approach trades-off accuracy for reduced costs and quicker times while guaranteeing budget and deadline requirements of queries, however, it does not provide accuracy guarantees during AaaS delivery that are essential to support reliable and accurate decision making.

There are related works that diversify the BDAAAs for the AaaS platforms. Agarwal et al. [140] propose BlinkDB to approximately process large datasets for reliable results with response time error bounds. Zhang et al. [57] build an ND-based solution to split large medical data in clouds providing faster responses with reduced costs and higher flexibility. Tordini et al. [56] process sequencing datasets of complex workflows applying data splitting techniques while maintaining high accuracy.

Mian et al. [74] apply heuristics to provision resources for data analytics workloads with effective resource configuration in a public cloud, however, they sacrifice SLAs to reduce resource costs. This is different from our SLA guarantee purpose as satisfactory AaaS delivery is the fundamental for user satisfaction. Wang et al. [103] support data-aware scheduling for efficient load balancing, however, they do not consider QoS, SLA guarantee, or cost optimization. Xia et al. [79] support a fair and collaborative way to place big data into distributed datacenters, however, they do not consider QoS, SLAs, or indeed AaaS delivery.

Garg et al. [75] manage resources for general cloud applications instead of big data analytics applications. Gu et al. [76] support cost-minimized big data analytics in distributed datacenters, however, does not support cost-optimization and SLA guarantees. Zheng et al. [90] proposed a variety of algorithms to schedule big data workflows on clouds to minimize the costs under deadlines. However, they do not provide automatic scheduling, nor consider budget constraints or consider admission control satisfying SLAs. Dai et al. [109] propose a multi-objective resource allocation approach for big data applications to obtain optimal deployment of VMs in clouds. However, they neither consider admission control nor tackle the time challenges for big data processing.

Zhou et al. [110] propose a declarative optimization engine to provision cloud resources for workflows utilizing available GPU power for timely solutions. They neither consider QoS requirements nor SLA guarantees. Mao et al. [86] propose an auto-scaling mechanism for traditional workflow scheduling to satisfy task deadlines while minimizing financial costs. However, their approach does not target big data analytics or consider budget or accuracy requirements of requests. Chen et al [80] propose a utility model-based scheduling mechanism for cloud service provider to optimize profits or user satisfaction. However, they do not consider budget, deadline, accuracy as the key QoS factors of tasks. They trade-off customer satisfaction while we aim to maximize user satisfaction by providing high quality AaaS satisfying query QoS requirements.

The novelty of our research work is proposing data sampling-based admission control and cloud resource scheduling algorithms with SLA guarantees on budget and deadline requirements to support big data analytics solutions under limited times and tight budgets. Moreover, we provide accuracy bounds for big data analytics results to

support reliable and accurate decisions. Our proposed algorithms are able to provide profit optimized and time minimized AaaS solutions for fast, cost-effective, and reliable problem solving and decision making in various BDAA domains, which cannot be supported by the above related works.

5.6 Conclusion and Future Works

Admission control and resource scheduling serve as the key functions to maximize profits and minimize query response times for AaaS platforms to deliver SLA-guaranteed AaaS to various domains of users. We proposed timely, cost-efficient, and reliable query admission control and resource scheduling solutions by modelling, formulating, and implementing the data sampling-based multi-objective optimization solutions. Experimental evaluation shows the admission control and scheduling algorithms significantly increase admission rates, increase profits, and reduce resource costs with efficient resource configurations for the entire the query workloads and for different BDAAs under real time and periodic scheduling scenarios compared to the state-of-the-art algorithms. Furthermore, our proposed algorithms significantly benefit big data analytics under tight deadlines and limited budgets by supporting fully accurate query processing as well as sampling-based query processing that enable users to obtain cost and time effective big data solutions with accuracy guaranteed AaaS for timely, cost-efficient, and reliable decision making.

As part of the future research works, we will continue investigating and proposing automatic and efficient optimization algorithms to tackle big data analytics challenges under various QoS requirements. We will keep working on proposing effective and efficient sampling and splitting-based optimization algorithms to help AaaS platforms to deliver satisfactory AaaS to various domains of users offering faster query response times and reduced cloud resource costs without compromising the reliability of big data analytics solutions.

6 SLA-Aware and Data Splitting Based Profit Optimization of Resource Management

This chapter addresses Problem Scenario 4 to support big data analytics in domains where rapid turnaround of results is required for reliable and rapid decision-making under hard time constraints. This chapter presents algorithms that process queries with full accuracy for reliable decision-making as needed by applications such as banking and emergency services. This chapter is derived from paper¹ and makes key contributions to resource management algorithms that maximize profits and minimize query times to enable reliable and fast results for deadline-constrained big data processing.

This chapter enhances the resource management solutions of Chapter 3 by enabling big data processing under hard deadlines. This chapter differs from Chapter 4 and Chapter 5 where accuracy of queries could be traded-off for quicker responses and reduced costs. To enable fast responses, the algorithms utilize SLA aware and data splitting-based methods to speed up data processing and guarantee SLAs based on budgets and time constraints. The algorithms apply auto-scaling and data locality-based scheduling with prioritized resource utilization to efficiently schedule resources and react to dynamic resource requirements to ensure the best data processing locality, optimized query times, and minimized resource costs whilst maximizing resource utilization. The scheduling algorithms offer flexible task dependencies to support queries with different task structures. Experimental evaluation shows that the proposed algorithms perform significantly better compared to state-of-the-art optimization algorithms allowing increasing query admission rates and higher profits, whilst supporting efficient resource configurations.

This chapter is originally derived from the following publication:

¹Y. Zhao, R. N. Calheiros, A. V. Vasilakos, J. Bailey, and R. O. Sinnott, “SLA-aware and deadline constrained profit optimization for cloud resource management in big data analytics-as-a-service platforms,” in *Proceedings of 2019 IEEE International Conference on Cloud Computing (CLOUD)*, pp. 146-155, 2019.

SLA-Aware and Deadline Constrained Profit Optimization for Cloud Resource Management in Big Data Analytics-as-a-Service Platforms

Yali Zhao¹, Rodrigo N. Calheiros², Athanasios V. Vasilakos³, James Bailey¹, Richard O. Sinnott¹

¹The University of Melbourne, Australia

²Western Sydney University, Australia

³Lulea University of Technology, Sweden

Abstract

Discovering optimal data analytics solutions to extract value from data for better and faster decision making is essential for many application domains, especially in the big data era. Big data analytics typically requires a tremendous amount of computational resources to process large data volumes that can be very expensive and time consuming. Our research focuses on providing optimization solutions for Analytics-as-a-Service (AaaS) platforms that automatically and elastically provision cloud resources to execute queries guaranteeing Service Level Agreements (SLAs) across a range of Quality of Service (QoS) requirements. We propose admission control and resource scheduling algorithms for AaaS platforms to maximize profits while providing time-minimized query execution plans to meet user demands and expectations. To enable timely responses as required for many domains, the algorithms utilize data splitting-based query admission and resource scheduling offering parallel processing on the split datasets. Extensive experiments are conducted to evaluate the algorithm performance compared to state-of-the-art optimization algorithms. Experimental results show that our algorithms perform significantly better from a range of perspectives including increasing query admission rates and creating higher profits, whilst

supporting efficient resource configurations that are able to support big data processing demands under tight deadlines.

Keywords: Profit Optimization, Service Level Agreement, Admission Control, Resource Scheduling, Analytics-as-a-Service, Data Splitting, Big Data, Cloud Computing.

6.1 Introduction

In the big data era, many researchers and scientists require optimal data analytics solutions to deal with large quantities of data, which are rapidly produced across many application domains. Big data is difficult to process and manage with traditional techniques and tools [146]. Analyzing big data to extract timely knowledge from such voluminous datasets is increasingly demanded for many application domains. Cloud computing provides a distributed computing paradigm that supports big data analytics with elastic resource provisioning delivered in a pay-as-you-go manner [130]. However, setting up and managing large-scale computing environments for big data analytics remains challenging for users without a background in distributed computing and related big data analytic technologies [11]. Big Data Analytics Applications (BDAAs) are typically designed to analyze data in specific application domains that can serve major research projects or large enterprises; however, these are typically not targeted to non-technical individuals or occasional usage scenarios [12].

Analytics-as-a-Service (AaaS) platforms leverage big data processing capabilities to deliver data analytics solutions for various domains and communities. These are offered as consumable services in a pay-as-you-go model and should ideally support easy-to-use mechanisms without targeted expertise required by end users. We aim to support the development of AaaS platform by proposing automatic and efficient resource management solutions that optimize profits for AaaS providers whilst enabling AaaS to be delivered at lower costs, with timely responses, and with SLA guarantees on QoS requirements.

In some disciplines, e.g. clinical and medical sciences, the accuracy of big data analytics is essential for reliable decisions that typically require processing of the full datasets. These disciplines may also require fast decision making where results need to

be returned in a timely manner. Efficiently analyzing big data for timely decisions remains a challenge. As the volume of datasets increases to a level that exceeds the storage and processing capabilities of existing computing resources, data splitting can be used to split larger datasets to smaller datasets for parallel processing to obtain timely and accurate data analytics results.

To enable the AaaS platform to deliver timely and reliable results, automatic and efficient query admission control and cloud resource scheduling are essential. Related research works such as [6, 11, 12, 13, 21, 46, 56, 74, 90, 96, 98, 107, 109] study resource scheduling problems in cloud computing environments. However, none of these works support data splitting-based admission control and cloud resource scheduling to support reliable and fast decision making with SLA guarantees when dealing with QoS requirements including deadlines and budgets.

Our **motivation** of this work is to provide automatic and efficient admission control and resource scheduling algorithms to optimize the profits of AaaS providers whilst minimizing query times and delivering deadline constrained AaaS solutions to users within given budgets. In tackling this, admission control and cloud resource scheduling of the AaaS platform must tackle the following research challenges:

- Queries are accompanied with different QoS requirements including budgets and deadlines. Requests can be stochastic with varying resource and application demands that require effective admission control to admit queries to ensure QoS can be met.
- SLA guaranteed AaaS delivery is important for user satisfaction, and hence it is important for AaaS platforms to have penalty-free and high-quality service delivery.
- To tackle tight deadline constrained big data processing on large datasets, data splitting techniques can be used to support parallel processing. These require adaptive admission control and resource scheduling algorithms.
- Deadline constrained multi-objective optimization to maximize profits and minimize response times under various constraints represents a complex optimization problem that requires accurate modelling and formulation.
- Well-designed mechanisms such as auto-scaling, data locality-based resource provisioning, and prioritized resource utilization needs to be applied to support effective and efficient resource scheduling and management.

To tackle the above challenges, we make the following **key contributions**:

- We formulate the multi-objective profit optimization and time minimization scheduling problem for reliable and fast big data processing under tight deadlines.
- We propose automatic and elastic multi-objective optimization admission control and resource scheduling algorithms that serve multiple optimization objectives while guaranteeing SLAs with regards to AaaS delivery.
- To tackle deadline constrained big data analytics challenges, the proposed algorithms apply data splitting-based methods that support parallel data processing for reliable and timely results.
- The scheduling algorithms apply auto-scaling and data locality-based scheduling with prioritized resource utilization. These mechanisms allow the AaaS platform to efficiently schedule resources and react to dynamic resource requirements to ensure the best data processing locality, optimized query times, and minimized resource costs whilst maximizing resource utilization.
- The scheduling algorithms enable flexible and complex task dependencies to support queries with different task structures. Data analytics requests can vary from independent tasks to complex dataflows.

We evaluate the performance of the proposed admission control and cloud resource scheduling algorithms through extensive experiments, which show the proposed algorithms outperform the state-of-the-art algorithms that is able to admit more queries, create higher profits, and provide more efficient large-scale resource configurations that enable timely solutions under tight deadlines and with controllable budgets.

6.2 Problem Statement

6.2.1 Problem Model

In this section, we present the key models that underpin the optimization problem.

Queries can vary from simple queries, such as an age enquiry of a patient database, to complex query such as machine learning algorithms for cancer genetics analytics. A query can contain QoS requirements related to budgets and deadlines; the requested resources; the *BDAA* used to execute the query; the dataset characteristics including the data type, distribution, locality, and size; and any task execution dependencies that define the structure of the query.

*BDAA*s brokered from third-party providers contain the information of application types and profiles. The *BDAA* profile details the *BDAA* costs, required resource times

including the data splitting, processing, and integration times, and the resource configurations required for different queries with given QoS requirements. Obtaining profiles related to BDAAAs from various domains requires expert knowledge, and hence reliable profiles are assumed to be provided by external application providers.

Resources contain resource types, such as individual servers or clusters of memory-optimized VMs; the resource capacities, such as the CPU, memory, storage, the number of resources for query execution, and the resource costs.

Costs involved in the AaaS delivery include the overall Resource Cost (RC); the BDAA Cost (BC) that is assumed to be a constant fee charged by application providers; the Penalty Cost (PC) if any SLA is violated for given data analytics requests; the overall AaaS Cost (AC) that are paid by users for using the AaaS, and the overall profit (PRO) created by the AaaS platform. AC charged for each query is assumed to be a fixed fee that is determined by the QoS requirements, resource utilization requirements, requested BDAAAs. Therefore, AC is a constant value. Big data is assumed to be stored in cloud data centers and the data transfer and storage costs are directly paid by users.

6.2.2 Problem Statement

The deadline constrained profit optimization problem to maximize the profits for AaaS providers and minimize the query response times under various constraints is an NP-complete decision problem, which can be polynomially formulated as a multi-objective optimization problem based on mixed Integer Linear Programming (ILP) [147] models. The profit of the AaaS platform is calculated as (1).

$$PRO = AC - RC - BC - PC \quad (1)$$

We aim to provide SLA guaranteed resource management solutions satisfying QoS requirements of queries to improve user satisfaction and avoid high penalty costs. Since AC and BC are constant while PC is zero with SLA guaranteed AaaS delivery, optimization of PRO can be transformed to minimization of RC while minimizing query response times with SLA guarantees.

The overall optimization Objective Z (4) is formulated as a combination of Objective X (2) and Objective Y (3) using the symbols defined in Table 6.1.

$$\textbf{Objective } X = \text{minimize}(\sum_{j=1}^n PR_j * C_j * t_j) \quad (2)$$

Table 6.1: Definition of Symbols

Symbol	Definition
i	The big data analytics request / query
j	The cloud resource
m	The set of queries
n	The set of cloud resources
t_j	The to-be-purchased time for j
T_j	The time that is available for j
LRT_j	The left-over resource time of j in the last schedule
RRT_{ij}	The required cloud resource time to process i on j
x_{ij}/x_{kj}	The scheduling decision of assigning i / k to j
PR_j	The resource utilization priority of j
s_i/s_k	The time when i / k starts execution
b_{ik}/b_{ki}	The query execution sequence of i and k
D_i	The QoS requirement of deadline for i
B_i	The QoS requirement of budget for i
C_{ij}	The execution cost of i using j
EAT_j	The earliest available time of j to execute queries
EPT_j	The purchased end time of j in the last schedule
TFT_j	The time when query finish execution of created j
CT_j	The current clock time of j
RCT_j	The time required to create cloud resource j
S_{ij}	The data splitting time of i on j
P_{ij}	The data processing time of i on j
I_{ij}	The data integration time of i on j
SC_{ij}	The data splitting cost of i using j
PC_{ij}	The data processing cost of i on j for the split dataset
IC_{ij}	The data integration cost of i on j for all split results

A resource is represented by j ; a set of cloud resources is represented by n ; the resource times to-be-purchased for j is represented by t_j ; the unit cost of j is represented by C_j ; the resource utilization priority of j is represented by PR_j and this

adds weights to less prioritized resources in the cost function to minimize the usage of such resources. *Objective X* minimizes the overall costs of utilizing different resources to deliver optimal AaaS solutions.

Resource utilization priorities are given to resources based on the following criteria: 1) earlier created resources, 2) cheaper resources, 3) single resources. Prioritized resource utilization aims to maximize resource utilization by prioritizing query scheduling using different resource types, and hence allow subsequent created, expensive, and/or clusters of resources with query workloads to be gradually reduced and ultimately terminated at periodic checkpoints. Priority-based resource utilization allows for the full utilization of computing resources whilst minimizing resource costs to align with the optimization objective.

$$\textbf{Objective } Y = \text{minimize}(\sum_{i=1}^m s_i) \quad (3)$$

A query is represented by i ; a set of queries is represented by m ; and start time of i is represented by s_i . *Objective Y* finds an optimal solution to minimize the costs with the best resource configurations to execute queries at the earliest starting time s_i to minimize t_j for better cost savings and higher user satisfaction.

$$\textbf{Objective } Z = \text{minimize}(F_0 * \sum_{j \in n} (PR_j * C_j * t_j) + \sum_{i \in m} s_i) \quad (4)$$

Objective Z is an integration of sub-objectives X and Y to minimize cloud resource costs while establishing a scheduling solution with minimized query times. Optimizing Z subject to Constraints (6-45) is a lexicographic optimization problem [121]. The importance of X contributing to the overall optimization problem is higher than Y . To ensure the aggregated optimization problem based on minimizing sub-objectives is consistent with the original profit optimization problem, coefficient F_0 (5) is assigned to X to ensure that changes in X dominate changes in Y .

$$F_0 = \max(Y) - \min(Y) + 1 \quad (5)$$

Query Resource Capacity Constraints (6) ensure the sum of the times of j used to execute queries should not exceed the available times of j . Constraint (7) determines $x_{ij} = 1$ when i is assigned to j while $x_{ij} = 0$ when i is not scheduled to j . RRT_j indicates the Required Resource Time (RRT) to analyze the complete dataset of i on j ; and t_j is the times to-be-purchased for j to execute queries. The value of T_j is set in (8). For a given created resource j , T_j is set as the Left Resource Time (LRT_j) from the previous schedule. For newly created resource j , T_j is set as $-RCT_j$ that excludes cloud Resource Creation Time (RCT_j) from t_j .

$$\sum_{i \in m, j \in n} (RRT_{ij} * x_{ij}) \leq t_j + T_j \quad (6)$$

$$x_{ij} = \begin{cases} 1, & i \text{ is assigned to } j \\ 0, & \text{otherwise} \end{cases}, \forall i \in m; j \in n \quad (7)$$

$$T_j = \begin{cases} LRT_j, & j \text{ created} \\ -RCT_j, & \text{otherwise} \end{cases}, \forall j \in n \quad (8)$$

Data Splitting-Based Query Resource Constraints (9) ensure that the sum of the data splitting time S_{ij} , data processing time P_{ij} , and data integration time I_{ij} should not exceed the available time of resource j .

$$\sum_{i \in m, j \in n} ((S_{ij} + P_{ij} + I_{ij}) * x_{ij}) \leq t_j + T_j \quad (9)$$

Query Execution Sequence Constraints (10) determine the unique query execution sequence needed to achieve the optimization solution. b_{ik} is a binary variable that specifies if request i is assigned to run in front of k . If request i is scheduled in front of request k , $b_{ik} = 1$; otherwise, $b_{ik} = 0$.

$$b_{ik} = \begin{cases} 1, & i \text{ is executed before } k \\ 0, & i \text{ is executed after } k \end{cases}, \forall i, k \in m \quad (10)$$

Data Splitting-Based Execution Sequence Constraints (11) define that data processing task i should not execute earlier than data splitting task k . Constraint (12) defines that a data integration task i can only start after all data processing tasks end execution. Data splitting is necessary when queries have tight deadline constraints or where no single resource can meet the query demand. Data splitting techniques generate smaller split datasets from the original larger datasets. Processing smaller split datasets in parallel can significantly reduce query times. An example dataflow related to data splitting-based query processing is shown in Figure 6.1.

$$b_{ik} = 1, \text{ if } i \text{ generates data splits for } k, \forall i, k \in m \quad (11)$$

$$b_{ki} = 1, \text{ if } i \text{ integrates split results of } k, \forall i, k \in m \quad (12)$$

Query Dependency Constraints (13) define the task structures of big data analytics requests that range from simple structures such as bag of tasks to complex structures such as enterprise workflows or scientific dataflows. A complex big data analytics request is composed by different individual tasks with given task logic and execution orders. When task i is a father task of k , $b_{ik} = 1$ ensures that a task k starts execution only when all dependent tasks end processing. These task dependency constraints enable the dynamic definition of task structures to enable the AaaS platform support analytics tasks with heterogeneous structures and dependencies in a flexible way.

$$b_{ki} = 1, \text{ if task } i \text{ is a child task of } k, \forall i, k \in m \quad (13)$$

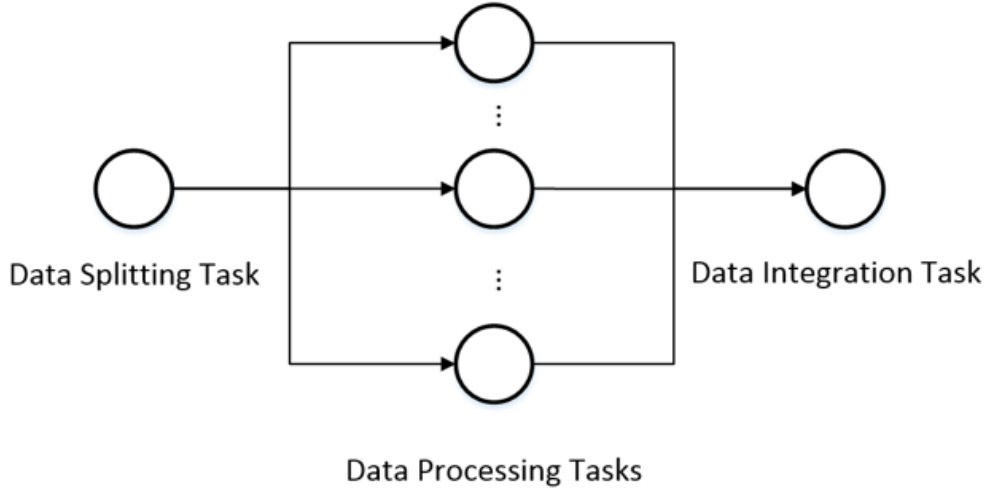


Figure 6.1: Dataflow Example for Splitting-Based Query Processing.

Available Resource Configuration Constraints (14) ensure that query requests can only be submitted to resources with sufficient capacity to run the queries without violating QoS, which only allows those resources with sufficient capacity required by queries to be considered by the ILP optimization solver. This significantly reduces the search space of the optimization problem and hence improves the efficiency and performance of the optimization.

$$x_{ij} = 0, \text{ if } j \text{ is not available to execute } i, \forall i \in m, j \in n \quad (14)$$

Data Locality Aware Constraints (15) guarantee that queries can only execute on resources where the datasets can be accessed to avoid lengthy transfer times and overheads. $x_{ij} = 0$ is used to ensure i will not be assigned to j if j has no access to the data. $x_{ij} = 1$ is set when such query assignment is consistent with the optimal resource scheduling solution.

$$x_{ij} = 0, \text{ if data of } i \text{ not be accessed by } j, \forall i \in m, j \in n \quad (15)$$

Query Scheduling Times Constraints (16) guarantee that the overall scheduling times x_{ij} for any query i on any resource j must be 1 to ensure SLAs while assigning i to run on one resource.

$$\sum_{i \in m, j \in n} x_{ij} = 1 \quad (16)$$

Query Budget Constraints (17) restrict the query cost of i to execute on j should be within user-specified budget B_i . Constraint (18) guarantees that if the resource cost of j exceeds the budget specified by query i , j will be removed from the search space to

support the ILP solver to find a timely solution in a reduced search space and hence improve the performance of the optimization algorithm.

$$C_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (17)$$

$$x_{ij} = 0, \text{ if } j \text{ violates budget of } i, \forall i \in m; j \in n \quad (18)$$

Data Splitting-Based Query Budget Constraints (19) restrict the overall costs of data splitting-based query processing on j should be within B_i . Data splitting-based query processing is used when the deadline of big data analytics request is tight that no existing resource can process the original dataset within given deadline. Data splitting enable parallel processing of the smaller split datasets to deliver timely AaaS solution. The overall query processing costs of i on resource j should meet the budget constraint B_i , as shown in (19), where SC_{ij} is the cost of splitting, PC_{ij} is the cost of processing i on the split datasets, and IC_{ij} is the cost of integrating all data processing results of the split datasets to generate the final results.

$$(SC_{ij} + PC_{ij} + IC_{ij}) * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (19)$$

Query Deadline Constraints ensure AaaS solutions are delivered before query deadlines. Constraint (20) guarantees the unique execution order of queries by specifying that only one of b_{ik} or b_{ki} can be 1. Constraint (21) ensures resources that cannot satisfy the deadline requirement of any query are eliminated from the solution space for efficient optimization solutions. Constraint (22) guarantees when i and k are scheduled to the same j , i should run either before k or after k . This is transformed from non-linear relationship (27) by linearization method. Constraint (27) guarantees that either $b_{ik} = 1, b_{ki} = 0$ when i executes before k or $b_{ik} = 0, b_{ki} = 1$ when i runs after k . Constraint (23) is generated from non-linear constraint (28) by the Big M method to guarantee that i should finish processing the full dataset before k starts execution when $b_{ik} = 1$. Here the F_1 has a sufficiently large constant value satisfying (29) [11]. Constraint (24) is transformed from non-linear constraint (30) by linearization. It ensures query i should end execution before its deadline D_i if it executes on resource j at s_i . F_2 is a constant satisfying (31). Constraint (25), derived from the non-linear constraint (32) through linearization, is used to guarantee that if i is scheduled on j then the finish time of i cannot exceed the purchased end time of j . This is given as the sum of EPT_j in the last schedule and the purchased new time t_j in the current schedule. F_3 is a constant that satisfies (33) with a sufficiently large value. Constraint (26) is transformed from the non-linear relationship (34) through linearization and ensures that i should start execution at

the Earliest Available Time (EAT) EAT_j when scheduled to resource j . F_4 is a constant that is sufficiently small satisfying (35). Constraint (36) ensures that for a created j , EAT_j is the Task Finish Time (TFT_j) when j has a query to run on it or the current Clock Time (CT_j) when j has no query to run on it; otherwise, if j is not created yet, EAT_j is calculated as the sum of CT_j and RCT_j .

$$b_{ik} + b_{ki} \leq 1, \forall i, k \in m \quad (20)$$

$$x_{ij} = 0, \text{ if } j \text{ violates deadline of } i, \forall i \in m; j \in n \quad (21)$$

$$b_{ik} + b_{ki} - x_{ij} - x_{kj} \geq -1, \forall i, k \in m; j \in n \quad (22)$$

$$s_i - s_k + F_1 * b_{ik} \leq F_1 - RRT_{ij}, \forall i, k \in m; j \in n \quad (23)$$

$$s_i + F_2 * x_{ij} \leq F_2 + D_i - RRT_{ij}, \forall i \in m, j \in n \quad (24)$$

$$s_i - t_j + F_3 * x_{ij} \leq F_3 - RRT_{ij} + EPT_j, \forall i \in m, j \in n \quad (25)$$

$$s_i + x_{ij} * F_4 \geq F_4 + EAT_j, \forall i \in m, j \in n \quad (26)$$

$$\left. \begin{matrix} x_{ij} = 1 \\ x_{kj} = 1 \end{matrix} \right\} \xrightarrow{\text{either}} \begin{cases} b_{ik} = 1, b_{ki} = 0 \\ b_{ik} = 0, b_{ki} = 1 \end{cases}, \forall i, k \in m, j \in n \quad (27)$$

$$b_{ik} = 1 \Rightarrow s_i + RRT_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (28)$$

$$F_1 \geq \max (s_i + RRT_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (29)$$

$$s_i * x_{ij} + RRT_{ij} \leq D_i, \forall i \in m, j \in n \quad (30)$$

$$F_2 \geq \max (s_i + RRT_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (31)$$

$$x_{ij} = 1 \Rightarrow s_i + RRT_{ij} \leq t_j + EPT_j, \forall i \in m, j \in n \quad (32)$$

$$F_3 \geq \max (s_i - t_j + RRT_{ij} - EPT_j) + 1, \forall i \in m, j \in n \quad (33)$$

$$x_{ij} = 1 \Rightarrow s_i \geq EAT_j, \forall i \in m, j \in n \quad (34)$$

$$F_4 \leq \min (s_i - EAT_j) - 1, \forall i \in m, j \in n \quad (35)$$

$$EAT_j = \begin{cases} TFT_j, & \text{query is running} \\ CT_j, & \text{otherwise} \end{cases}, j \text{ is created}, \forall j \in n \quad (36)$$

$$\begin{cases} CT_j + RCT_j, & \text{otherwise} \end{cases}$$

Data Splitting-Based Query Deadline Constraints (37) are transformed from the non-linear relationship (40) by linearization guarantees that query i should finish data splitting, processing, and integration before k starts if $b_{ik} = 1$. F_5 has a constant value that is sufficiently large and satisfies (41). Constraint (38) is transformed from the non-linear relationship (42) by linearization and ensures i should finish splitting and execution no later than its deadline D_i after it starts execution on j at s_i . F_6 has a constant value that is sufficiently large satisfying (43). Constraint (39) is transformed from the non-linear relationship (44) through linearization. It ensures that the sum of the data splitting time S_{ij} , the processing time P_{ij} , and the integration time I_{ij} of i cannot

exceed the purchased time of j , which sums up the EPT_j in the previous schedule and t_j in the current schedule. A sufficiently large constant is assigned to F_7 satisfying (45).

$$s_i - s_k + F_5 * b_{ik} \leq F_5 - S_{ij} - P_{ij} - I_{ij}, \forall i, k \in m; j \in n \quad (37)$$

$$s_i + F_6 * x_{ij} \leq F_6 + D_i - S_{ij} - P_{ij} - I_{ij}, \forall i \in m, j \in n \quad (38)$$

$$s_i - t_j + F_7 * x_{ij} \leq F_7 - S_{ij} - P_{ij} - I_{ij} + EPT_j, \forall i \in m, j \in n \quad (39)$$

$$b_{ik} = 1 \Rightarrow s_i + S_{ij} + P_{ij} + I_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (40)$$

$$F_5 \geq \max (s_i + S_{ij} + P_{ij} + I_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (41)$$

$$s_i * x_{ij} + S_{ij} + P_{ij} + I_{ij} \leq D_i, \forall i \in m, j \in n \quad (42)$$

$$F_6 \geq \max (s_i + S_{ij} + P_{ij} + I_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (43)$$

$$x_{ij} = 1 \Rightarrow s_i + S_{ij} + P_{ij} + I_{ij} \leq t_j + EPT_j, \forall i \in m, j \in n \quad (44)$$

$$F_7 \geq \max (s_i - t_j + S_{ij} + P_{ij} + I_{ij} - EPT_j) + 1, \forall i \in m, j \in n \quad (45)$$

6.3 Admission Control and Resource Scheduling

Deadline constrained big data analytics requests cannot always be admitted and executed by AaaS platforms. To tackle this challenge, data splitting-based query admission control and resource scheduling algorithms can be applied to parallelly process split datasets and hence return results in a timely manner.

6.3.1 Admission Control

Admission control focuses on maximizing admission rates with SLA guarantees for admitted queries. This is important for higher profits and better user experiences. Effective query admission decisions should only admit queries satisfying QoS requirements by processing queries using existing BDAAAs based on the available resources and their configurations. Reliable admission control is the foundation for resource schedulers to generate query scheduling solutions with SLA guarantees. Queries that are admitted and scheduled by the AaaS platform include the following two scenarios. *Scenario 1*: queries with sufficient budgets and deadlines that enable full query processing on the entire dataset. *Scenario 2*: queries with tight deadlines that require fully accurate results such that data splitting is used for timely solutions using parallel query processing of the split datasets. The pseudocode of the query admission control algorithm is presented in *Algorithm 1*.

Users submit queries to the AaaS platform with dynamic submission times. The admission controller iteratively admits each query. It first checks whether the requested BDAA is available or if such a BDAA can be brokered from an external BDAA provider and whether the dataset of the query can be accessed. If both conditions are met, it determines if there is any configuration that can support the SLA-guaranteed query execution through searching in the resource registry where all resources are registered for the available Resource Configurations (*resourceConfigurations*). The admission controller determines the Estimated Cost (EC) and Estimated Execution Time (EET) for each configuration to execute the request at the earliest available time. If such a configuration exists, the *originQuery* that processes the full dataset is admitted as in *Scenario 1*. (Lines 1-10).

If tight deadline limits the query being processed in full, the query is then considered for data splitting-based admission (Lines 11-26). The admission controller estimates whether applying the data splitting-based admission method can meet the deadline of such a query by re-estimating the overall big data analytics cost as the sum of the data Splitting Cost (SC), the Processing Cost (PC), and the Integration Cost (IC). It calculates the overall query time as the sum of the data Splitting Time (ST), the data Processing Time (PT), and the data Integration Time (IT). If the splitting-based method supports requests being processed within its budget and deadline requirements, such a query is admitted as a *splittingQuery* as considered by *Scenario 2*. All requests that are rejected after the data splitting-based admission method are provided with feedback to pass on to users for modification of the original query specifications to support query resubmission (Lines 27-30).

Algorithm 1: Query Admission Control Algorithm

Inputs: big data analytics requests submitted by users (submittedQueries)

Outputs: query admission results or query rejection decisions

- 1: **for** each big data analytics request $\in$ submittedQueries
- 2: BDAA $\leftarrow$ search the application registry for requested BDAA
- 3: **if** the BDAA is brokerable / available && data can be accessed
- 4: resourceConfigurations $\leftarrow$ search registry of the cloud resources
- 5: **for** each resourceConfiguration $\in$ resourceConfigurations
- 6: EET = estimate execution time of the query
- 7: EC = estimate the resource cost of executing the query
- 8: **if** EET < queryDeadline && EC < queryBudget

```
9:      originQueries  $\leftarrow$  admit the query to process it in full
10:    end if
11:    else apply data splitting for parallel processing
12:      for each not admitted queries
13:        SC  $\leftarrow$  estimate cost applying data splitting
14:        PC  $\leftarrow$  estimate cloud resource cost to process data splits
15:        IC  $\leftarrow$  estimate cloud resource cost processing data integration
16:        ST  $\leftarrow$  estimate data splitting time
17:        PT  $\leftarrow$  estimate data processing time
18:        IT  $\leftarrow$  estimate data integration time
19:        if SC+PC+IC < queryBudget & ST+PT+IT < queryDeadline
20:          splittingQueries  $\leftarrow$  admit the request to process data splitting
21:          avaiRes  $\leftarrow$  update the available resources
22:        end if
23:      end for
24:    end else
25:  end for
26: end if
27: if the query cannot be admitted as originQuery or splittingQuery
28:   reject the query and detail the rejection reasons to the user
29: end if
30: end for
```

6.3.2 Resource Scheduling

1. *SPplitting-Based Profit Optimization (SPPO)*

SPPO supports SLA-aware and deadline constrained resource scheduling by applying data splitting-based methods that support resource allocations to run queries on smaller split datasets to meet given SLAs. This is especially helpful for AaaS platforms when tight deadlines restrict the AaaS delivery. SPPO supports full query processing on entire datasets as well as parallel query processing on split datasets for timely results with controllable costs. The data splitting-based algorithm is data splitting technique independent. Different data processing techniques can be efficiently utilized as required depending on the problem scenarios as well as the data analytics requirements. Profiles of BDAAAs contain resource configurations, data processing, splitting, and integration times and costs of the data splitting methods. These are required as inputs to the optimization scheduler so that the scheduler is able to make timely and reliable SLA-based profit optimization decisions. The pseudocode for the SPPO is detailed in *Algorithm 2*.

Users submit queries to the AaaS platform specifying the BDAAAs to analyze the full datasets. For every requested BDAA, the SPPO scheduler calls the admission controller to get the *originQueries* whose SLA can be guaranteed by processing the full dataset as well as *splittingQueries* that potentially split the full dataset for parallel processing under tight deadlines as *anaQueries*. If such queries exist, the SPPO scheduler gets all existing resources of the BDAA as *avaiRes*, along with the current resource configuration as input. SPPO applies heuristics to determine the input resources needed by the ILP solver by calling the *selectSplitResHeu* method (Lines 1-7). SPPO then applies *SPPOScheduling* to provide profit-maximization and cost-minimization scheduling solutions for optimization *Objective Z*. Afterwards, the SPPO scheduler adopts the feasible scheduling solution with the help of a solution interpretation function (Lines 8-11). Otherwise, the heuristic scheduling approach *maxDelayTime* is used to adopt the heuristic solution with the help of solution interpretation function *interpretAndAdoptHeuristicSolution* (Lines 12-15). Afterwards, the SPPO scheduler conducts periodic checks to scale resources down to save resource costs (Lines 16-19). Future scheduling is triggered when users submit more query requests to the AaaS platform.

The *SPPOScheduling* method (Lines 20-26) obtains up-to-date information in the AaaS platform including BDAAAs, query requests, and cloud resources needed as input to the optimization ILP solver. Afterwards, SPPO scheduler defines the *Objective Z* subject to Constraints (6-45) for the optimization problem. Finally, SPPO generate feasible optimization resource scheduling results within the given scheduling limits and these are utilized as the profit optimized resource scheduling solutions.

Algorithm 2: SPPO Cloud Resource Scheduling Algorithm

Inputs: admitted big data analytics requests, requested BDAAAs, cloud resources, resource configurations

Outputs: cloud resource scheduling solutions to execute admitted queries

- 1: **for** each query requested BDAA $\in$ all available BDAAAs
- 2: $originQueries \leftarrow$ get data analytics requests on original datasets
- 3: $splittingQueries \leftarrow$ get data splitting query requests
- 4: $anaQueries \leftarrow originQueries + splittingQueries$
- 5: **if** ($anaQueries > 0$)
- 6: $avaiRes \leftarrow$ update existing available resources of BDAA
- 7: $heuRes \leftarrow selectSplitResHeu(avaiRes, anaQueries, BDAA)$

```
8:    optimalSolution  $\leftarrow$  SPPOScheduling (anaQueries, heuRes, BDAA)
9:    if any feasible scheduling solution is returned within timeout
10:        interpretAndAdoptOptimalSolution (optimalSolution)
11:    end if
12:    else
13:        heuristicSolution  $\leftarrow$  maxDelayTime (heuRes, anaQueries, BDAA)
14:        interpretAndAdoptHeuristicSolution (heuristicSolution)
15:    end else
16:    res  $\leftarrow$  update existing available cloud resources of BDAA
17:    scaleResourcesDown (res)
18: end if
19: end for
20: Method: SPPOScheduling (anaQueries, heuRes, BDAA)
21:    updateInfo (anaQueries, heuRes, BDAA)
22:    setILPObjective (Objective Z)
23:    setILPConstraints (Constraints 6-45)
24:    setTimeOut (timeout)
25:    solveILPProblem ()
26:    optimalSolution  $\leftarrow$  getILPSolution ()
27: Method: selectSplitResHeu (avaiRes, anaQueries, BDAA)
28:    exeRes  $\leftarrow$  select resources from avaiRes
29:    newRes  $\leftarrow$  generate new resources based on MDT heuristic
30:    heuRes  $\leftarrow$  exeRes + newRes
31: Method: maxDelayTime (anaQueries, heuRes, BDAA)
32:    sort anaQueries based on maximized delay times
33:    assign anaQueries to resources with minimized EST
34:    heuristicSolution  $\leftarrow$  getHeuristicSolution ()
```

The *selectSplitResHeu* method (Lines 27-30) generates heuristic resources *heuRes* as input to the *SPPOScheduling* method. It first determines from *avaiRes* that can at least satisfy the QoS requirements of one query as *exeResources*. Following this, it utilizes the *maxDelayTime* based scheduling method to decide on the newly required resources *newRes* by mapping *anaQueries* to *exeRes* at the Earliest Start Time (EST). The *selectResHeu* method combines *exeRes* and *newRes* as *heuRes* to provision near optimal resources to allow *SPPOScheduling* to generate timely solutions. This significantly reduces the algorithm runtime of *SPPOScheduling* by reducing the search space for efficient optimization. The *maxDelayTime* method (Lines 31-34) is utilized to generate a heuristic scheduling solution as the backup algorithm to enforce SLAs when optimal solutions are not returned within given scheduling time limits.

2. Profit Optimization Scheduling (PO) Algorithm

The PO scheduling algorithm [11] applies SLA-based scheduling mechanisms to optimize profits for AaaS providers while minimizing processing times of data analytics requests with guaranteed SLAs. The PO scheduling algorithm is suitable as a comparison algorithm as it serves the profit optimization objectives of the AaaS platform as SPPO. PO also applies mixed ILP programming model to provide optimal scheduling solutions to enable the AaaS platform to maximize profits and deliver SLA guaranteed solutions with minimized query times with full accuracy.

PO is subject to different constraints: Resource Capacity Constraints, Query Deadline Constraints, Query Budget Constraints, Query Scheduling Times Constraints, and Data Locality Constraints. PO is able to enable resource scheduling for full query processing under sufficient budgets and deadlines; however, PO is not able to support big data analytics with tight deadlines that can be achieved with data splitting and parallel processing. SPPO is able to support data splitting-based resource scheduling by coordinating various new constraints to enable tight deadline constrained big data processing while PO does not support such features. Both PO and SPPO adopt various heuristics as backup solutions for unexpectedly large query arrivals that exceed the capacity for the ILP solver to return data analytics results with SLA guaranteed AaaS delivery for high user satisfaction.

6.4 Performance Evaluation

We conducted extensive experiments to analyze the efficiency and performance of the proposed admission control and resource scheduling algorithms for query admission control, SLA guarantee, profit optimization, cost saving, resource configuration, and algorithm running time.

6.4.1 Experiment Setup

1. Experiment Environment: To evaluate the efficiency of the proposed query admission control and cloud resource scheduling algorithms, we built the AaaS framework as experimental environment utilizing CloudSim [122]. CloudSim supports modeling and simulation of distributed cloud computing environments offering on-demand cloud resource provisioning and configuration. This allows conducting repeatable and controllable experiments to evaluate the performance of the proposed

algorithms without having to pay for potentially expensive cloud computing resources and associated BDAA costs. For the optimization, IBM CPLEX 5.5 [6.9] is utilized as the ILP solver. This provides an optimization decision support toolkit that enables time-efficient development, validation, and deployment of optimization solutions through constraint-based programming and mathematical models. Extensive experiments were conducted for the algorithm performance evaluation for real time scheduling where queries were scheduled based on arbitrary arrival times and periodic scheduling where queries were scheduled based on particular periodic Scheduling Intervals (SIs) with units specified in minutes.

2. *Resource Configuration:* We simulated four Amazon Web Service (AWS) datacenters that each contained 500 physical nodes. Every node consisted of 400 CPU cores, 30TB memory, 10PB storage, and 10GB/s bandwidth. Based on the Amazon Elastic Compute Cloud (EC2) VM model, we simulated six types of memory optimized VMs: r4.large, r4.xlarge, r4.2xlarge, r4.4xlarge, r4.8xlarge and r4.16xlarge [123]. The possible cloud resource configurations to execute queries varied from CPU cores, individual VMs to clusters of VMs. The cloud resources were charged based on the utilized times in dollars.

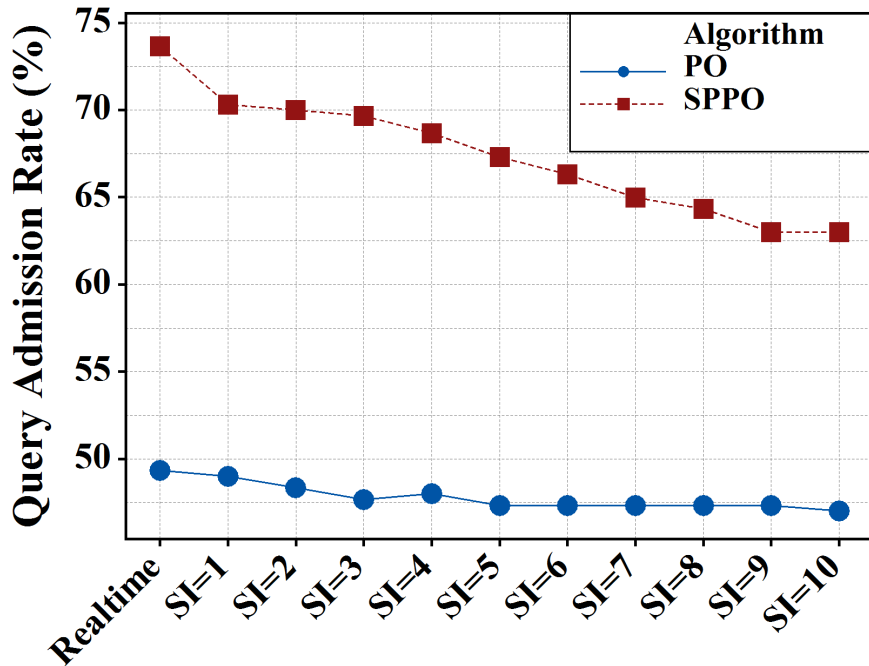


Figure 6.2: Admission Rates of SPPO and PO.

3. *Big Data Analytics Workload* was generated using Big Data Benchmark [124] and Normal Distribution (ND) splitting techniques [57]. The ND technique generates smaller split datasets to explore data splitting on big data with tight deadlines. The big data benchmark provides data analytics times and cloud resource configurations that run queries on large datasets in the Amazon EC2 Cloud. It utilizes big data analytic frameworks given information including data sizes, types and locations. The ND splitting workload details the cloud resource configurations and query response times needed to split and process big data. Based on the above information, the times and costs are generated for data splitting and query processing on full datasets and split datasets using different BDAAs in various configurations.

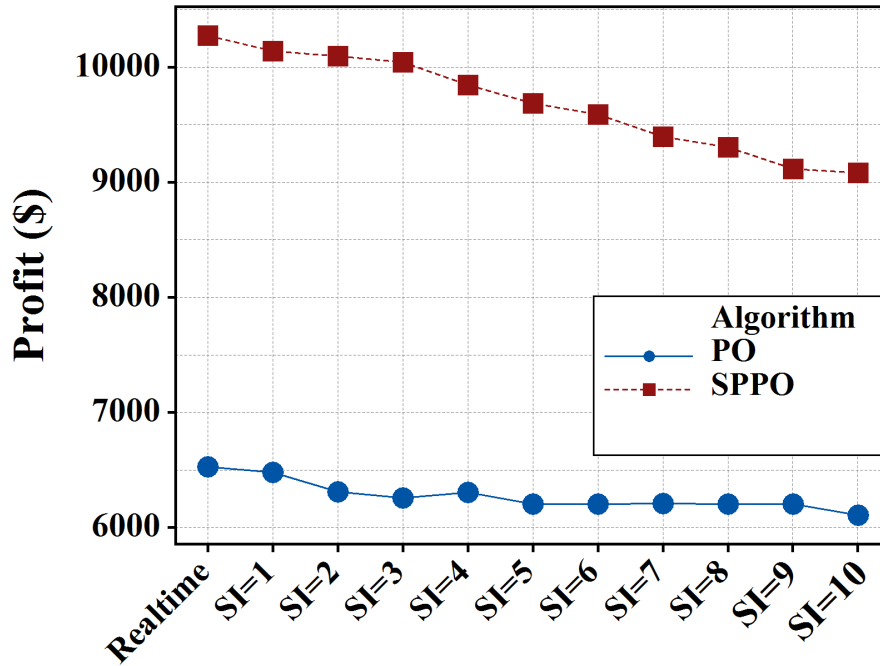


Figure 6.3: Profits of SPPO and PO.

4. *Big Data Analytics Requests* include the following information: *Submission times* utilize a one-minute mean Poisson arrival; the *Types* of queries include scan, join, aggregation, and user defined functions; *BDAA*s include Hive on Hadoop (BDAA₁), Hive on Spark (without caching) (BDAA₂), and Hive on Spark (with caching) (BDAA₃); *Resource times* include details of the requirements of the cloud computing resources needed to execute the queries; *Users* submit queries to utilize the AaaS; *Deadlines* contain tight deadlines based on ND (3, 1.4) and loose deadlines based on ND (8, 3)

[148]; *Budgets* contain tight budgets based on ND (3, 1.4) and loose budgets based on ND (8, 3) [148].

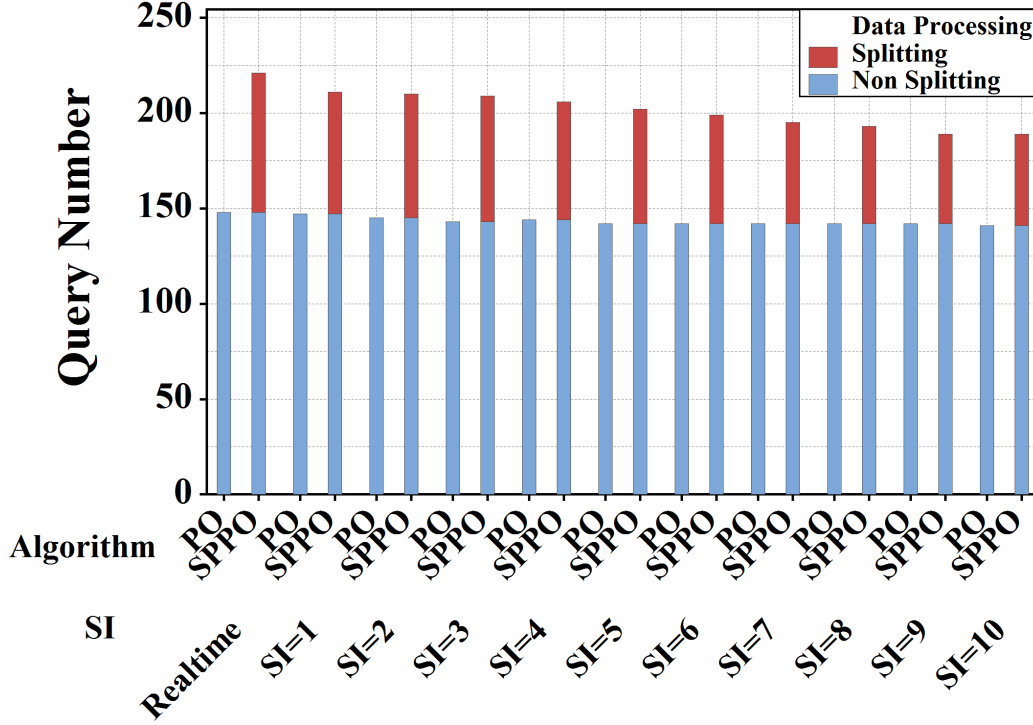


Figure 6.4: Query Processing Methods of SPPO and PO.

6.4.2 SLA-Aware and Data Splitting-Based Scheduling

1. Query Admission Control and SLA Guarantees

To evaluate the efficiency and effectiveness of admission control, we conducted experiments for real time resource scheduling and periodic resource scheduling with SIs varied from 1-10 minutes. We compared the algorithm performance between SPPO and PO with the results shown in Figure 6.2. We can see that SPPO is able to admit more queries for both real time and periodic resource scheduling with significantly increased query admission rates between [16-24%]. Results show the significant performance advantages of SPPO with higher admission rates that can generate higher profits and improve user satisfaction by accommodating more analytics requests for processing. Moreover, the results also show the effectiveness of admission control by supporting 100% SLA guaranteed query processing.

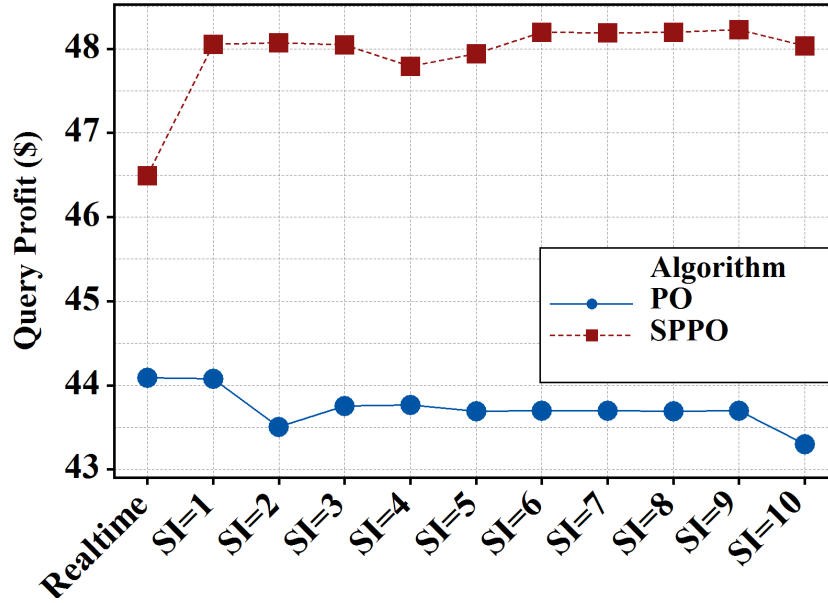


Figure 6.5: Query Profits of SPPO and PO.

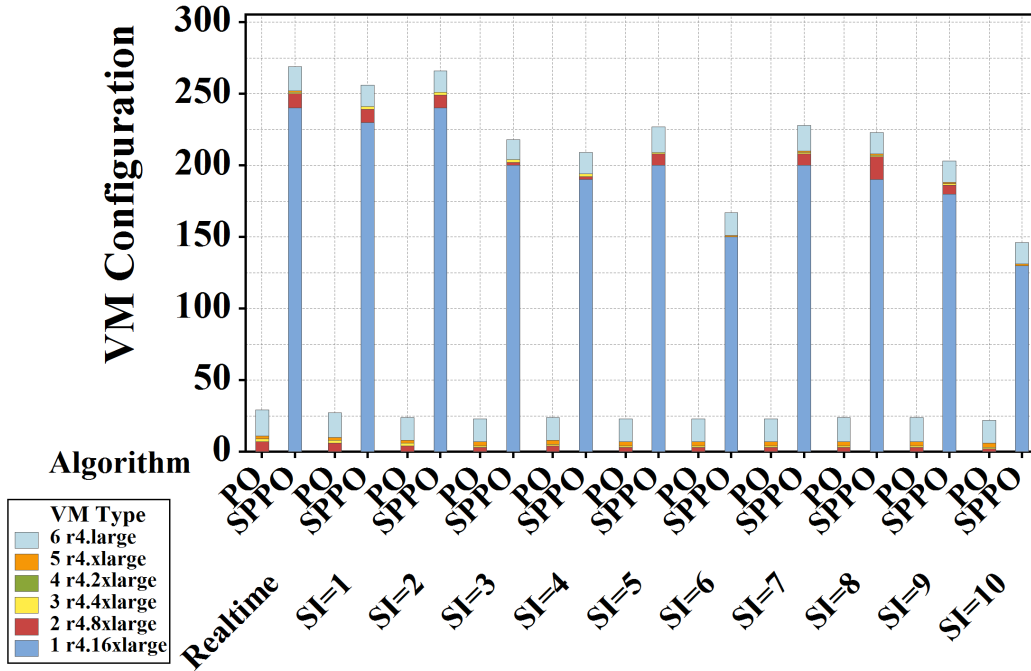


Figure 6.6: VM Configuration of SPPO and PO.

2. Profit Enhancement

The profit enhancement advantages of SPPO were evaluated for real time and periodic scheduling scenarios. The results in Figure 6.3 show that SPPO can create significantly higher profits for AaaS providers than PO for all scheduling scenarios.

The increased profit ranges between [47-60%]. The enhanced profits stem from the higher query admission rates and increased utilization of resources with better resource configuration and query scheduling solutions when higher number of queries are scheduled by the optimization solver. We also notice an overall decreasing trend for both SPPO and PO due to queries being rejected where they have tight deadlines for longer periodic SIs.

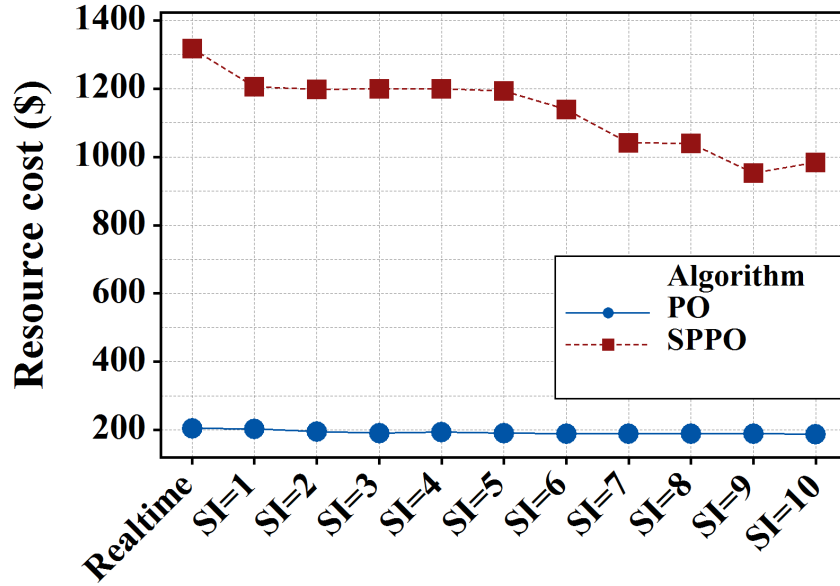


Figure 6.7: Resource Costs of SPPO and PO.

It is not always appropriate to apply real time scheduling in AaaS platforms due to the significant resource consumption incurred with frequent scheduling. This also decreases the performance advantages of the optimization algorithm. SI=1 was selected as the suitable periodic SI that could admit more queries and create higher profits, and hence it was selected as the SI for the AaaS platform. For simplicity purposes, we utilize SI=1 as the periodic SI for performance analysis for the periodic scheduling scenarios.

We analyzed the data processing methods of SPPO and PO to better understand the performance advantages of SPPO with results shown in Figure 6.4. We can see that SPPO supports both full processing of the original large datasets and data splitting-based processing of the small split datasets under tight deadlines, while PO cannot tackle tight deadline challenges of big data processing that can only support full query processing under sufficient budgets and deadlines. The higher number of queries processed using the data splitting-based methods for SPPO are in the range [47-73],

which is in the percentage interval of [16-24%] from 300 submitted queries. This demonstrates increased query admission rates that offer higher profits for the AaaS platform by applying SPPO as the data splitting-based optimization algorithm.

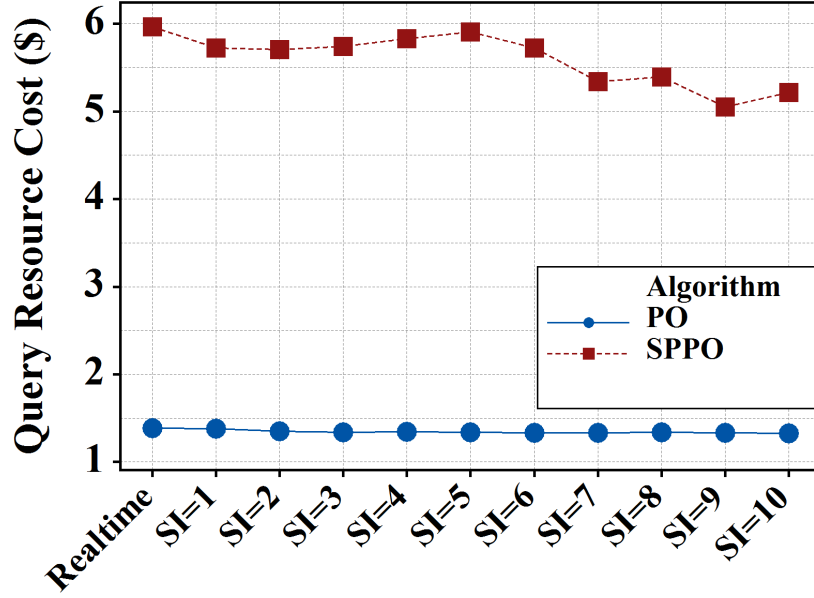


Figure 6.8: Query Resource Costs of SPPO and PO.

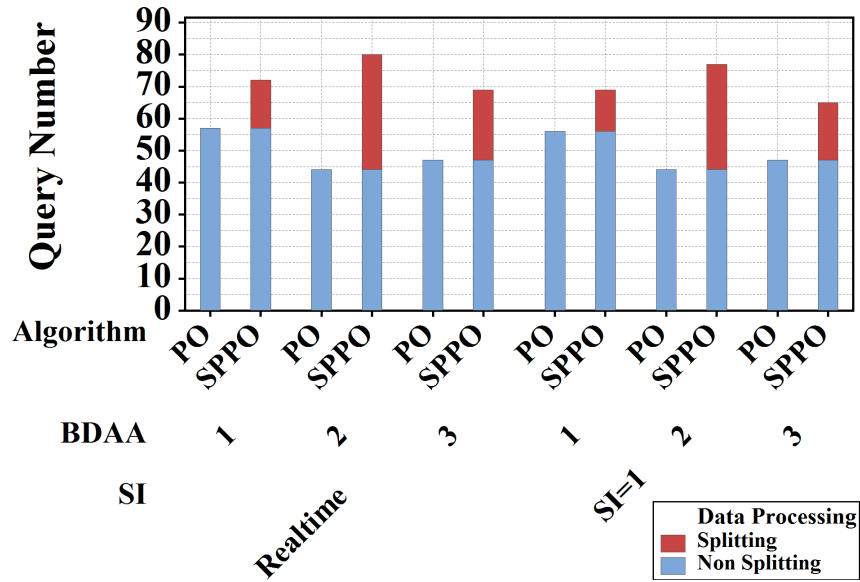


Figure 6.9: Query Number for Splitting and Non Splitting Processing of SPPO and PO for BDAA.

We further analyzed the average query profits of SPPO compared to PO. Figure 6.5 shows that the average query profits of SPPO are increased compared to PO ranging

between [5-11%]. This demonstrates that SPPO is able to enhance profits whilst maintaining performance advantages even with larger SIs and associated higher query rejection rates.

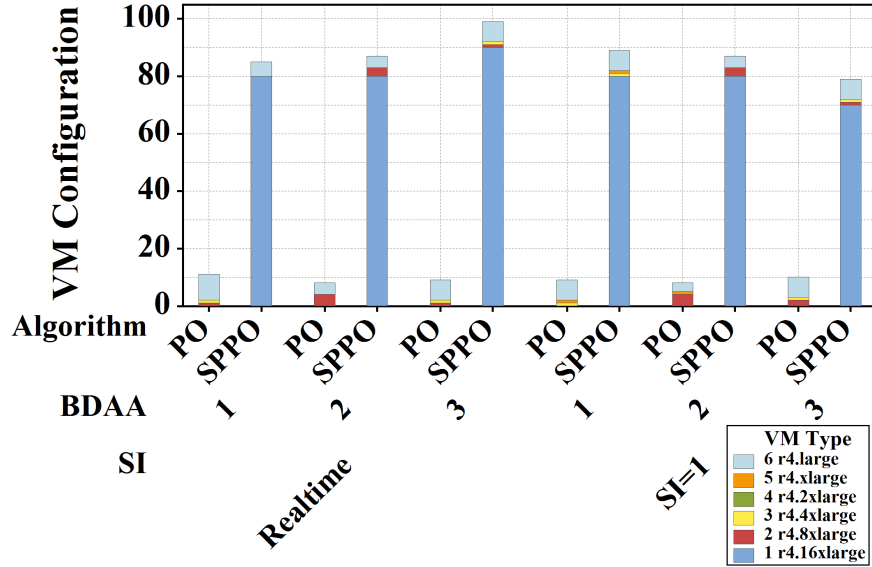


Figure 6.10: Resource Configurations of SPPO and PO for BDAA.

3. Cost Saving and Resource Configuration

The resource configurations of SPPO and PO for real time scheduling and periodic scheduling are shown in Figure 6.6. We can see the resources utilized for the VM types. As seen, there is a trend whereby VMs of r4.16large are more frequently utilized by SPPO. As shown in Figure 6.7, we can see that the resource costs of SPPO is 4-5 times the resource costs of PO while the average resource cost per query is 5-6 times that of PO as shown in Figure 6.8. This stems from the higher number of queries that are processed and increased resource requirements of big data processing under given deadlines. This is consistent with timely response requirements of big data analytics requests. In order to deliver query results in a timely manner when processing huge volumes of data, parallel processing is required on split datasets. This consumes an increased amount of resources. r4.16xlarge is able to provide higher computing capacity to satisfy tight query deadlines, therefore r4.16xlarge are utilized more frequently with the usage interval in the range [130-240] for SPPO. As seen, the SPPO algorithm also has a higher number of resources being utilized for all VM types due to

the higher computing resource requirements for the data analytics workloads under tight deadlines.

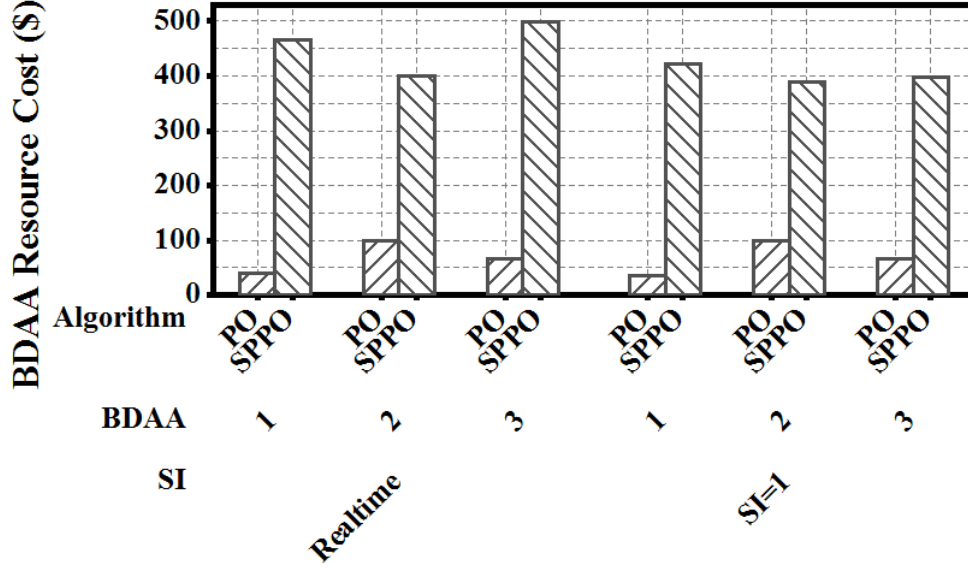


Figure 6.11: BDAA Resource Costs of SSPO and PO.

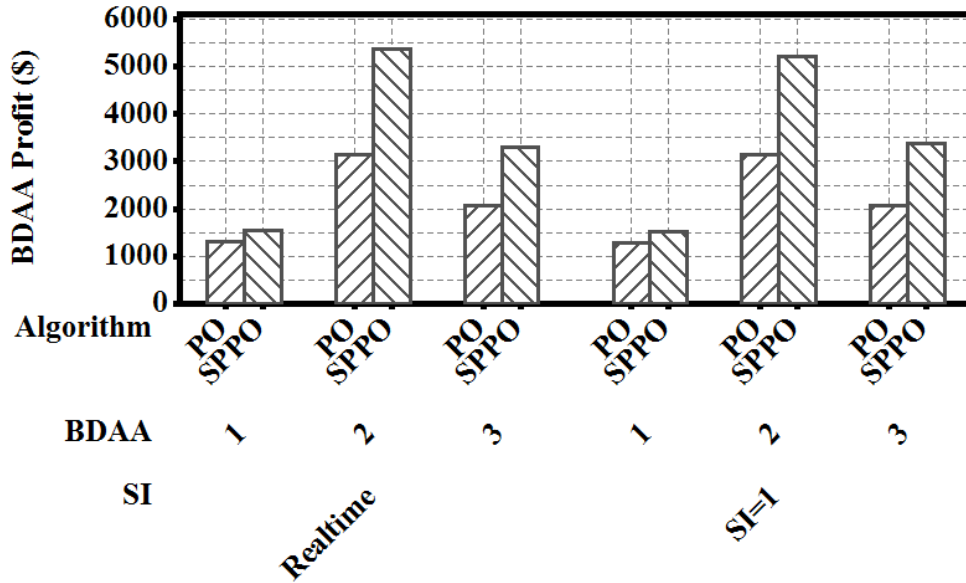


Figure 6.12: BDAA Profits of SPPO and PO.

4. Algorithm Performance for Different BDAAs

We also consider the performance analysis for every BDAA to evaluate the performance advantages of SPPO. We use real time scheduling and SI=1 as the example to illustrate the performance advantages of SPPO. The results in Figure 6.9 show that

SPPO can admit more requests with an increased admission rates between [23-75%] for data splitting and non-splitting based query processing for both real time and SI=1. The results show that SPPO is able to support higher query admission and profit enhancement for all BDAA's and it outperforms PO for both real time and periodic scheduling.

We further analyze the resource configurations of SPPO and PO. Results in Figure 6.10 show that SPPO consumes significantly more resources than PO due to higher query workloads that consume increased computing resources. The results show that VMs r4.16xlarge have significant higher number of resource utilization that is in range of [70-90]. This is caused by the increased resource consumption demands for big data processing under tight deadlines. Further analytics of resource costs per BDAA indicates that the resource costs of SPPO are 3-11 times that of the resource costs of PO for BDAA 1-3, as shown in Figure 6.11. This is consistent with larger resource requirements for big data processing for increased number of accepted queries with tight deadlines. Such configurations show the capacity of SPPO for large scale computing for big data processing under tight deadlines. The profits created by SPPO for BDAA 1-3 are [17-71%] higher than PO as shown in Figure 6.12. The results show that SPPO is able to facilitate big data processing with large-scale configurations to tackle tight deadline constraints while significantly increasing profits with higher admissions.

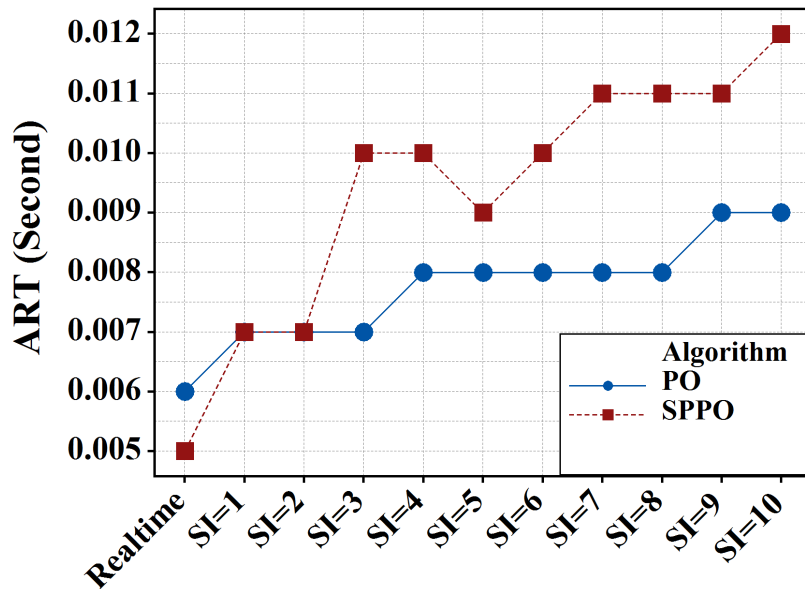


Figure 6.13: ART of SPPO and PO.

5. *Algorithm Running Time (ART) Analysis*

The average ART of SPPO and PO is shown in Figure 6.13. We see an overall increasing trend of ART due to more queries being scheduled and more resources being consumed that results in more complex optimization problems under various constraints. The results show that both SPPO and PO can deliver scheduling solutions within 5-12 milliseconds. This illustrates the efficiency of the optimization algorithms. Therefore, the ART is not a limiting factor for the proposed optimization algorithms. We also designed a range of heuristic scheduling algorithms to deal with unexpectedly large number of query arrivals that might cause delays in optimization solution delivery and ensuring SLAs are satisfied.

All of the obtained results show that SPPO significantly outperforms PO under numerous performance indicators including higher admission rates, higher profits with efficient resource configurations needed to meet the QoS requirements of SLAs for large scale and realistic query workloads. SPPO outperforms PO for each BDAA for both real time and periodic scheduling. The results show that SPPO is able to tackle big data processing challenges under tight deadlines by providing timely solutions suitable for large-scale parallel processing.

6.5 Related Work

AaaS platforms support and benefit better decision making by delivery AaaS with SLA guarantees for individuals and organizations in various application domains. We focus on supporting the AaaS platform with elastic and automatic admission control and cloud resource scheduling algorithms. We compare our work with leading related works.

Zhao et al. [6] support admission control and two-phase resource scheduling algorithms to enhance profits for AaaS platforms by delivering cost-effective cloud resource scheduling solutions where the solutions can be sub-optimal. Zhao et al. [11] propose a profit-optimization scheduling algorithm to maximize profits for AaaS platforms while minimizing the query responses and guaranteeing SLAs for full query processing under sufficient deadlines and budgets. Zhao et al. [13] propose a data splitting-based query admission and resource scheduling solution. This is used to explore big data processing challenges under limited budgets and tight deadlines where accuracy can be traded-off against reduced costs and faster response times by

processing samples that are representatives of the original datasets. It is not suitable for BDAA domains that require full accuracy or for reliable decision making as we contribute in this paper by supporting parallel processing of the full big data applying splitting-based algorithms for timely and reliable AaaS.

Sun et al. [12] provision cost-effective solutions for multi-tenancy scenarios as part of an analytic framework without considering query admission or SLAs. Mian et al. [74] provide heuristic algorithms to find cost-effective cloud resource provisioning and configuration for data analytics for given applications in a public cloud. They schedule queries to trade-off SLAs for cheaper costs, which differs from our focus of SLA guarantee. Tang et al. [98] put forward a self-adaptive resource scheduling algorithm to reduce MapReduce start times. Zulkernine et al. [21] leverage a conceptual AaaS architecture that does not consider admission control. Sfrent et al. [107] propose many task scheduling to optimize the number of processed tasks in big data platforms without considering SLA guarantees. Zhang et al. [96] leverage multi-objective scheduling to find a near optimal solution to reduce times for many tasks. Dai et al. [109] obtain optimal deployment solutions for big data applications using a multi-objective VM allocation approach. However, they do not consider admission control nor deadline constraints for processing big data. Zheng et al. [90] put forward resource scheduling algorithms to execute big data workflows to minimize the costs satisfying deadline constraints. However, they do not support admission control, automatic scheduling, or SLA guarantees.

Zhang et al. [57] support a ND based big data splitting technique for cloud-based medical data processing. This approach offers higher flexibility for big data processing with reduced cloud resource costs and faster query response times. Tordini et al. [56] process workflows whilst maintaining a high level of accuracy using data splitting techniques. Wong et al. [46] support Thrifty to deliver a Database-as-a-Service for lower costs with parallel query processing and SLA guarantees. These technologies enable AaaS platforms to provide diversified big data analytics solutions for various application domains.

None of the above works consider SLA-aware and data splitting-based query admission and resource scheduling solutions suitable for big data domains, and especially those domains that require reliable and rapid decision making. This is the novel contribution put forward in the paper.

6.6 Conclusion and Future Works

Our research focuses on providing data analytics solutions that maximize profits for AaaS platforms, whilst minimizing query times for users with guaranteed SLAs through automatic and efficient query admission control and associated cloud resource scheduling. Such solutions are needed to tackle challenges of big data processing under tight deadlines. We modeled and formulated a SLA-aware and data splitting-based multi-objective optimization problem. We proposed an admission control algorithm that provided reliable and effective query admission control solutions and a profit-optimized and time-minimized resource scheduling algorithm (SPPO). We demonstrated that the proposed algorithms were able to provide timely solutions with minimized query times whilst ensuring SLA guarantees. Experimental evaluation showed that the proposed algorithms significantly increased query admission rates and profits, and reduced resource costs for data analytics workloads supporting different BDAAAs under real time and periodic resource scheduling scenarios.

As part of our future research, we intend to improve the discovery and support of efficient and automatic AaaS solutions. This will include data splitting-based admission control and resource scheduling algorithms. We shall also explore other factors that can impact on job scheduling, e.g. security and privacy of big data processing. This is especially important in domains such as health, where privacy and data security are paramount.

7 SLA-Aware Data Splitting and Sampling Based Profit Optimization of Resource Management

This chapter is derived from paper¹ and addresses Problem Scenario 5 to support different big data processing scenarios where data analytic requests have heterogeneous budgets, deadlines, and accuracy constraints. This chapter improves the work in Chapter 3-6 by supporting optimal resource management solutions in the following ways: 1) processing queries with sufficient deadlines and budgets so that full datasets can be processed under budget and deadline constraints with accurate results for reliable decision making; 2) processing queries with sufficient budgets but limited deadlines, such that data splitting can be applied for parallel data processing to tackle tight deadline challenges with significantly reduced query times, and 3) processing queries with limited budgets and deadlines, where data sampling needs to be applied to approximately process data samples for fast and cost-effective results to tackle big data processing under limited cost and time constraints.

To process big data under insufficient deadlines and budgets, the algorithms allow data splitting and data sampling-based algorithms to reduce data processing times and resource costs based on parallel and approximate data processing. Experimental results show that the proposed algorithms outperform the state-of-the-art algorithms by significantly increasing query admissions, enhancing profits, reducing resource costs, providing large-scale resource configurations while delivering AaaS with SLA guarantees for the complete (accurate) query workloads, for all BDAAAs, and with real time scheduling and periodic scheduling.

This chapter is originally derived from the following publication:

¹ Y. Zhao, A.V. Vasilakos, J. Bailey, and R.O. Sinnott, “Profit Optimization for Splitting and Sampling Based Resource Management in Big Data Analytics-as-a-Service Platforms in Cloud Computing Environments,” in *Proceedings of International Conference on e-Science (e-Science)*, pp. 156-167, 2019.

Profit Optimization for Splitting and Sampling Based Resource Management in Big Data Analytics-as-a-Service Platforms in Cloud Computing Environments

Yali Zhao¹, Athanasios V. Vasilakos², James Bailey¹, Richard O. Sinnott¹

¹The University of Melbourne, Australia

yalizhao.alice@gmail.com, baileyj@unimelb.edu.au, rsinnott@unimelb.edu.au

²Lulea University of Technology, Sweden

athanasios.vasilakos@ltu.se

Abstract

Exploring optimal big data analytics solutions to benefit various domains in decision making and problem solving becomes an ever-important research area. Big data Analytics-as-a-Service (AaaS) platforms offer online AaaS to various domains in a pay-as-you-go model. Big data analytics incurs expensive costs and takes lengthy processing times due to large-scale computing requirements. To tackle the cost and time challenges for big data analytics, we focus on proposing efficient and automatic resource management algorithms to maximize profits and minimize query times while guaranteeing Service Level Agreements (SLAs) on Quality of Service (QoS) requirements of queries. For query processing constrained by tight deadlines and limited budgets, our proposed algorithms enable data splitting and sampling-based resource scheduling for parallel and approximate processing that significantly reduce data processing times and resource costs. We formulate the multi-objective resource scheduling problem to optimize profits for AaaS platforms while guaranteeing SLAs of queries with minimized response times. We design extensive experiments for algorithm performance evaluation, results show our proposed algorithms outperform state-of-the-art algorithms that maximize profits for AaaS platforms while improving admission rates and minimizing response times for queries. The scheduling algorithms support elastic and automatic

large-scale resource configurations to minimize resource costs, and deliver timely, cost-effective, and reliable AaaS with SLA guarantees.

Keywords- Profit Optimization, Resource Scheduling, Big Data, Sampling, Splitting, Analytics-as-a-Service, Service Level Agreement, Cloud Computing.

7.1 Introduction

Discovering optimal big data [149] solutions to guide decision making keeps motivating researchers and scientists to create values from big data. As we step into big data era, extremely large volume of structured, unstructured, and semi-structured data is generated at an ever-accelerating speed that exceeds the management and processing capacity of traditional techniques. Cloud computing [150] greatly benefits big data analytics [151] by provisioning configurable resources at infrastructure, platform, and application levels in a pay-as-you-go and on-demand way. Providing optimal solutions to extract insights from big data becomes essential for problem solving and decision making in various domains such as information technology and finance.

We propose the concept of a general-purpose cloud-based big data Analytics-as-a-Service (AaaS) platform [6] to serve on-demand big data analytics requests and provision AaaS to various domains of users as automatic services with the ease of utilization and at lower cost. The AaaS platform delivers AaaS at the platform level and brokers Big Data Analytics Applications (BDAA) from third-party BDAA providers and computing resources from infrastructure resource providers. The AaaS platform significantly reduces the AaaS costs that enable users only pay for AaaS they use without the need to purchase expensive hardware and software. This significantly benefits individuals or small enterprises for occasional usage. The AaaS platform greatly eases AaaS delivery by hiding the complexity of resource management for users without expertise in distributed resource managements. We focus on queries on large datasets leaving out data security and consistency issues. Service Level Agreements (SLAs) are established to meet Quality of Service (QoS) requirements.

For big data analytics, time, cost, and accuracy are key factors that significantly affect decision making and problem solving. Processing big data can be very expensive and time consuming due to large-scale computing requirements [11]. Due to different

requirements of application domains, some queries need full accuracy while other can trade-off query accuracy to reduce resource costs and query response times. Data splitting reduces query times by parallel processing but incurs higher costs due to additional data splitting and integration operations and hence suits queries with full accuracy requirements under tight deadlines and with sufficient budgets. For reliable decision making, we give priority to full accuracy and only trade-off accuracy where budgets and/or deadlines are limiting factors by applying data sampling while such processing is meaningful and with confidential intervals of accuracies. When time challenges can be tackled by either splitting or sampling, data splitting is given higher priority for fully accurate query processing.

A number of related works [6, 7, 11, 13, 22, 81, 85, 89, 91, 95, 104, 112, 113] study resource management with support of different techniques and algorithms. However, none of these works can provide cost-effective, reliable, and fast AaaS by applying splitting and sampling based optimal resource scheduling solutions with budget, deadline, and accuracy guarantees on service delivery, where comes the novelty of our work. Our research aims to optimize the platform profits while guaranteeing SLAs for AaaS with minimized times to support big data processing under limited budgets and tight deadlines. To achieve this, resource scheduling plays the key role to make optimal decisions to guide SLA-guaranteed query execution and resource provision.

To provide a comprehensive solution for timely, reliable, and cost-effective AaaS delivery, we focus on proposing novel resource scheduling algorithms to support the following query types and big data scenarios: (1) Sufficient budgets and deadlines: *FullQueries* are processed on the original datasets with full accuracy and optimized resource configurations. (2) Tight deadlines and sufficient budgets: *SplitQueries* are processed on split datasets to speed up data processing by applying effective data splitting [57] with fully accurate results given when data analytics exceeds the processing capacity of existing resources. (3) Tight budgets and/or tight deadlines: *SampleQueries* are processed by effective approximate processing [152] on data samples for reduced query times and resource costs with accuracy guarantees on query results.

Our **motivation** is to provide automatic and efficient resource scheduling algorithms for AaaS platforms to tackle the following research challenges:

- SLA guaranteed profit optimization resource management subject to various constraints is a complex multi-objective optimization problem that requires accurate modelling and formulation.
- Queries have varied QoS requirements that should be satisfied by efficient data splitting and sampling based scheduling for profit enhancements and SLA guarantees.
- The dynamisms of arrival time, type, QoS, BDAA, datasets, task dependencies, and resource requirements of queries require automatic scheduling to optimally scale resources with SLA-guaranteed configurations.
- To face the time and cost challenges of big data analytics, data splitting and sampling based scheduling is required for optimized resource management with timely and cost-efficient resource provision and reliable AaaS solution.

The **key contributions** of our work are:

- We formulate the optimization problem for cloud resource scheduling by applying mixed Integer Linear Programming (ILP) model [118] satisfying various QoS requirements based on constraint programming.
- We propose efficient and automatic splitting and sampling based scheduling algorithms that maximize profits for AaaS platform while guaranteeing query QoS requirements on deadlines, budgets, and accuracies.
- We propose optimization scheduling algorithms that apply auto-scaling, prioritized resource utilization, data locality-aware, multi-dependency based mechanisms to serve the multiple optimization objectives.
- We propose splitting and sampling based scheduling to provision resources for *FullQueries*, *SplitQueries*, and *SampleQueries* that support fast and reliable AaaS under various big data analytics scenarios.

To achieve this, we formulate the multi-objective optimization resource management problem, refine the architecture and model of the AaaS platform, enhance the AaaS framework, implement a range of resource scheduling algorithms, and perform extensive experiments to evaluate algorithm performances. Evaluation shows our proposed algorithms outperform the state-of-the-arts algorithms by optimal resource management solutions with higher request admission rates, less resource costs, higher profits, and automatic and large-scale resource configurations. Our work supports a comprehensive resource management solution to tackle big data analytics challenges

that is able to deliver profit-maximized, time-minimized, and SLA-guaranteed AaaS solutions to support cost-efficient, timely, and reliable problem solving and decision making in various domains.

7.2 Problem Statement

The AaaS platform processes queries submitted by users requesting BDAA from various application domains. A query can vary from scan query to complex data analytics algorithms that contain requirements of BDAA, requested resources, QoS, data characteristics including size, type, distribution, and locality, and task dependencies. We assume datasets are stored in cloud datacenters where the storage costs are directly paid by users. We allocate cloud resources to datacenters where data resides to avoid lengthy times and expensive costs to transfer big data.

BDAAAs are brokered from various domains with application specifications and profiles including mappings of the resources to processing times, sampling times, splitting times of queries; configurations; and BDAA costs. Reliable application profiles are assumed to be provided by external BDAA providers as heterogeneous BDAA profiles requires experts with domain knowledge [13]. A resource contains the information of resource type such as individual or a cluster of Virtual Machines (VMs); the number of resources, the resource costs; and capacities such as storage, CPU, memory, and bandwidth.

The costs involved in the AaaS platform are: Resource Cost (RC) that charged for cloud resources, BDAA Cost (BC) charged by third-party BDAA providers, Penalty Cost (PC) compensated to users when SLA is violated, AaaS Cost (AC) paid by users for utilizing AaaS, and Profit (PR) created by the AaaS platform. We assume fixed BDAA costs, BC , are charged by BDAA provider and constant AaaS service fees, AC , are charged for specific query that is determined by the requested BDAA, QoS requirements of queries, and required cloud resources.

The multi-objective resource scheduling problem to optimize profits and minimize the query times with SLA guarantees subject to various constraints is an NP-complete problem, which can be polynomially reduced to a mixed Integer Linear Problem [118]. The symbols of the optimization model are defined in Table 7.1.

Profit $PR = AC - BC - PC - RC$. Since AC and BC are constants and PC is zero as long as SLAs are guaranteed, the optimization of PR is minimization of RC with

SLA guaranteed and query time minimized scheduling solutions.

Objective A = $\text{minimize } \sum_{j=1}^n (U_j * C_j * t_j)$, where j is a cloud resource; n is a cloud resource set; t_j is the leased times of j ; C_j is the unit cost to utilize j , and U_j is the priority of utilizing resource j . A aims to minimize RC that is calculated as the product of the utilization priority U_j of cloud resources, to-be-used resource time t_j , and the unit resource cost C_j . U_j prioritizes the resource utilization by assigning a lighter value for higher-prioritized resources including earlier created resources, cheaper resources, and single resources to benefit full utilization of resources and cost minimization by auto-scaling.

Objective B = $\text{minimize } (\sum_{i=1}^m s_i)$, where i is a query; m is a query set while s_i is the start time of request i . Objective B provides time minimized query execution and resource configurations to run queries at the earliest time to minimize t_j for better user experiences and cost optimization.

Objective Z combines A and B to minimize resource costs while minimizing query times subject to Constraints (1-51) formulates a lexicographic optimization problem [121]. A contributes higher importance to the optimization problem than B . Coefficient $F_0 = \max(B) - \min(B) + 1$ of A is used to ensure the aggregated optimization problem to minimize all individual objectives are consistent with the original profit maximization problem that any change in A should dominate all of the changes in B [120].

$$\text{Objective Z} = \text{minimize } (F_0 * \sum_{j \in n} (U_j * C_j * t_j) + \sum_{i \in m} s_i)$$

Query Assignment Constraints (1) define if query i is assigned to resource j , where x_{ij} is a binary variable. If i is assigned to resource j ; $x_{ij} = 1$; otherwise, $x_{ij} = 0$.

$$x_{ij} = \begin{cases} 1, & \text{if } i \text{ executes on } j \\ 0, & \text{otherwise} \end{cases}, \forall i \in m; j \in n \quad (1)$$

FullQuery Resource Constraints (2) ensure the sum of the resource times of FullQueries should be satisfied by the available times j . R_j is the query time to execute i on the complete dataset utilizing j , t_j is the hours to be utilized on j , T_j is the time remained in the last schedule on resource j . For new cloud resources, T_j has a negative value to deduct the creation time of resource j from t_j .

$$\sum_{i \in m, j \in n} (R_{ij} * x_{ij}) \leq t_j + T_j \quad (2)$$

Table 7.1: Symbol Definitions

Symbol	Definition
i	The big data analytics request / query
j	The cloud resource
m	The set of queries
n	The set of cloud resources
U_j	The resource utilization priority of j
C_j	The unit cost to utilize j
t_j	The to-be-purchased time for j to execute queries
s_i/s_k	The time when i / k starts execution
x_{ij}/x_{kj}	The scheduling decision of assigning i / k to j
b_{ik}/b_{ki}	The query sequence of i and k
R_{ij}	The time to execute i on the complete dataset using j
T_j	The resource time remained in the previous schedule on j
A_{ij}	The sampling time to process data samples on j
P_{ij}	The data splitting time of i on j
Q_{ij}	The query processing time of i on j
I_{ij}	The data integration time of i on j
D_i	The QoS requirement of deadline for i
E_j	The end of leased / purchased time in the last schedule
EA_j	The earliest available time of j
C_{ij}	The resource cost of i using j
B_i	The budget requirement of i
SC_{ij}	The data sampling cost of i using j to process data samples
PC_{ij}	The data splitting cost of i using j to split the large dataset
QC_{ij}	The data processing cost of i using j to process split datasets
IC_{ij}	The data integration cost of i using j to integrate results

SampleQuery Resource Constraints (3) ensure that the sampling time, A_{ij} , to obtain and process data samples from the original datasets utilizing j should be provided by the available resource time of j .

$$\sum_{i \in m, j \in n} (A_{ij} * x_{ij}) \leq t_j + T_j \quad (3)$$

SplitQuery Resource Constraints (4) guarantee that the sum of splitting time P_{ij} , processing time Q_{ij} , and integration time I_{ij} should be satisfied by the available time of j .

$$\sum_{i \in m, j \in n} ((P_{ij} + Q_{ij} + I_{ij}) * x_{ij}) \leq t_j + T_j \quad (4)$$

Available Resource Configuration Constraints (5) define $x_{ij} = 0$ if the resource configuration of j has no sufficient capacity to execute at least one query satisfying SLAs, the type of resources is eliminated from the search space. This greatly reduces the problem space of the solver and hence improves the efficiency of the optimization.

$$x_{ij} = 0, \text{ if } j \text{ is unavailable to execute } i, \forall i \in m, j \in n. \quad (5)$$

Query Structure Constraints (6) define task structures of queries including simple task structure, i.e., a bag of tasks, or more complex task structure, i.e., data workflows. $b_{ik} = 1$ when query i is the child task of query k that has to execute after k returns results.

$$b_{ik} = 1, \text{ if } k \text{ is a child task of } i, \forall i, k \in m \quad (6)$$

Query Execution Order Constraints (7) control the task execution order, determined by the optimized resource scheduling decision, where $b_{ik} = 1$ when i executes before k , otherwise, $b_{ik} = 0$.

$$b_{ik} = \begin{cases} 1, & i \text{ is scheduled before } k \\ 0, & \text{otherwise} \end{cases}, \forall i, k \in m \quad (7)$$

Query Unique Execution Order Constraints (8) ensure all queries including FullQueries, SampleQueries, and SplitQueries are executed in the order defined by the optimal scheduling solution allowing maximum one of b_{ik} or b_{ki} be 1.

$$b_{ik} + b_{ki} \leq 1, \forall i, k \in m \quad (8)$$

SampleQuery Execution Order Constraints (9) ensure query i to process data samples can only be executed after k that generates data samples.

$$b_{ki} = 1, \text{ if } i \text{ process sample of } k, \forall i, k \in m \quad (9)$$

SplitQuery Execution Order Constraints are used to ensure if splitting of big data is needed where big data analytics tasks have tight deadlines or no single resource can meet the computing demands for SplitQueries, constraint (10) defines data processing request i need to run after splitting task k while constraint (11) ensure data integration task i can only start after all of its father task k finishes data processing.

$$b_{ki} = 1, \text{ if } i \text{ generate data splits for } k, \forall i, k \in m \quad (10)$$

$$b_{ki} = 1, \text{ if } i \text{ integrate split results of } k, \forall i, k \in m \quad (11)$$

Query Scheduling Time Constraints (12) guarantee query i must be scheduled one time to one resource j and maximum one resource for execution to ensure SLAs are guaranteed.

$$\sum_{i \in m, j \in n} x_{ij} = 1 \quad (12)$$

Data Locality Constraints (13) define $x_{ij} = 0$ where resource j cannot access the dataset of query i to guarantee that queries should only run on cloud resources that have access to the datasets to tackle expensive big data transfer costs and lengthy overheads and reduces the search spaces of the optimization solver for time efficiency.

$$x_{ij} = 0, \text{ if } j \text{ cannot access data of } i, \forall i \in m, j \in n \quad (13)$$

FullQuery Deadline Constraints: Constraint (14) is transformed from non-linear constraint (23) by linearization that defines the execution order of i and k . Constraint (23) defines if both i and k are assigned to j , i either executes before k , defined as $b_{ik} = 1, b_{ki} = 0$; or executes after k , defined as $b_{ik} = 0, b_{ki} = 1$. Constraint (15) is transformed from non-linear constraint (24) through the Big M method to ensure if when $b_{ik} = 1$, task i needs to finish before k starts. F_1 has constant value that is sufficient large satisfying (16). Constraint (17), transformed from non-linear constraint (25), guarantees if query i starts on j at s_i , i has to end before the deadline, D_i . F_2 has constant value that is sufficient large satisfying (18). Constraint (19), transformed from the non-linear constraint (26), ensures if i is scheduled to j , it should finish execution before the purchased time of j , which is the sum of end time of lease E_j in the last schedule and the new time of lease t_j . F_3 has constant value that sufficient large satisfying (20). Constraint (21), transformed from the non-linear relationship (27), ensures i start at the earliest available time of resource j as EA_j . If j is created with query executing on it, EA_j is the end time of i on j ; otherwise, EA_j is the current time of the platform. If j is not created, EA_j is the sum of current time of the platform and resource creation time of j . F_4 has constant value that is sufficient small satisfying (22).

$$b_{ik} + b_{ki} - x_{ij} - x_{kj} \geq -1, \forall i, k \in m; j \in n \quad (14)$$

$$s_i - s_k + F_1 * b_{ik} \leq F_1 - R_{ij}, \forall i, k \in m; j \in n \quad (15)$$

$$F_1 \geq \max (s_i + R_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (16)$$

$$s_i + F_2 * x_{ij} \leq F_2 + D_i - R_{ij}, \forall i \in m, j \in n \quad (17)$$

$$F_2 \geq \max (s_i + R_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (18)$$

$$s_i - t_j + F_3 * x_{ij} \leq F_3 - R_{ij} + E_j, \forall i \in m, j \in n \quad (19)$$

$$F_3 \geq \max (s_i - t_j + R_{ij} - E_j) + 1, \forall i \in m, j \in n \quad (20)$$

$$s_i + x_{ij} * F_4 \geq F_4 + EA_j, \forall i \in m, j \in n \quad (21)$$

$$F_4 \leq \min(s_i - EA_j) - 1, \forall i \in m, j \in n \quad (22)$$

$$\begin{matrix} x_{ij} = 1 \\ x_{kj} = 1 \end{matrix} \} \xRightarrow{\text{either}} \begin{cases} b_{ik} = 1, b_{ki} = 0 \\ b_{ik} = 0, b_{ki} = 1 \end{cases}, \forall i, k \in m, j \in n \quad (23)$$

$$b_{ik} = 1 \Rightarrow s_i + R_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (24)$$

$$s_i * x_{ij} + R_{ij} \leq D_i, \forall i \in m, j \in n \quad (25)$$

$$x_{ij} = 1 \Rightarrow s_i + R_{ij} \leq t_j + EP_j, \forall i \in m, j \in n \quad (26)$$

$$x_{ij} = 1 \Rightarrow s_i \geq EA_j, \forall i \in m, j \in n \quad (27)$$

Deadline-Based Available Resource Constraints (28) guarantee that j without a resource configuration meeting the deadline requirements of any query including FullQuery, SampleQuery, and SplitQuery will be excluded from the resource selection pool to allow optimized scheduling solution to be returned timely in the reduced search space.

$$x_{ij} = 0, \text{ if } j \text{ violate deadline of } i, \forall i \in m; j \in n \quad (28)$$

SampleQuery Deadline Constraints: For SampleQuery, Constraint (29) ensure if $b_{ik} = 1$, query i should complete all of its data sampling and execution tasks on data sample before k starts execution. It is transformed from the non-linear relationship (35). A_i is the approximately processing of query i on data sample using resource j , F_5 has constant value that is sufficient large satisfying (30). Constraint (31) ensure that if i starts on j at time s_i , it should complete its sampling and processing before D_i , which is transformed from non-linear relationship (36). F_6 has constant value that is sufficient large obeying (32). If i is assigned to j , Constraint (33) ensures the data sampling and approximate processing time A_i finishes no later than the leased time of j as the sum of end time of previous lease, E_j , and new lease time, t_j . It is transformed from the non-linear relationship (37). F_7 has constant value that is sufficient large based on constraint (34).

$$s_i - s_k + F_5 * b_{ik} \leq F_5 - A_{ij}, \forall i, k \in m; j \in n \quad (29)$$

$$F_5 \geq \max (s_i + A_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (30)$$

$$s_i + F_6 * x_{ij} \leq F_6 + D_i - A_{ij}, \forall i \in m, j \in n \quad (31)$$

$$F_6 \geq \max (s_i + A_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (32)$$

$$s_i - t_j + F_7 * x_{ij} \leq F_7 - A_{ij} + E_j, \forall i \in m, j \in n \quad (33)$$

$$F_7 \geq \max (s_i - t_j + A_{ij} - E_j) + 1, \forall i \in m, j \in n \quad (34)$$

$$b_{ik} = 1 \Rightarrow s_i + A_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (35)$$

$$s_i * x_{ij} + A_{ij} \leq D_i, \forall i \in m, j \in n \quad (36)$$

$$x_{ij} = 1 \Rightarrow s_i + A_{ij} \leq t_j + E_j, \forall i \in m, j \in n \quad (37)$$

SplitQuery Deadline Constraints: For SplitQuery, Constraint (38) ensures that if $b_{ik} = 1$, query i should end data splitting, processing, and integration tasks before k starts execution as transformed from non-linear relationship (44). F_8 is a constraint that is sufficient large as defined by (39). Constraint (40) guarantee that if i runs on j at s_i , its completion time as sum of P_{ij} , Q_{ij} , and I_{ij} should be no later than its deadline D_i , as transformed from non-linear relationship (45). F_9 has constant value that is sufficient large satisfying constraint (41). Constraint (42) is transformed from non-linear constraint (46) that guarantees if i is assigned to j , its execution time including data splitting time (P_j), query processing (Q_j), and integration times (I_j) cannot exceed the leased time of j that sums up the remained resource time E_j in previous schedule and new leased time t_j . F_{10} is a sufficient large that satisfies constraint (43).

$$s_i - s_k + F_8 * b_{ik} \leq F_8 - P_{ij} - Q_{ij} - I_{ij}, \forall i, k \in m; j \in n \quad (38)$$

$$F_8 \geq \max (s_i + P_{ij} + Q_{ij} + I_{ij} - s_k) + 1, \forall i, k \in m, j \in n \quad (39)$$

$$s_i + F_9 * x_{ij} \leq F_9 + D_i - P_{ij} - Q_{ij} - I_{ij}, \forall i \in m, j \in n \quad (40)$$

$$F_9 \geq \max (s_i + P_{ij} + Q_{ij} + I_{ij} - D_i) + 1, \forall i \in m, j \in n \quad (41)$$

$$s_i - t_j + F_{10} * x_{ij} \leq F_{10} - P_{ij} - Q_{ij} - I_{ij} + E_j, \forall i \in m, j \in n \quad (42)$$

$$F_{10} \geq \max (s_i - t_j + P_{ij} + Q_{ij} + I_{ij} - E_j) + 1, \forall i \in m, j \in n \quad (43)$$

$$b_{ik} = 1 \Rightarrow s_i + P_{ij} + Q_{ij} + I_{ij} \leq s_k, \forall i, k \in m, j \in n \quad (44)$$

$$s_i * x_{ij} + P_{ij} + Q_{ij} + I_{ij} \leq D_i, \forall i \in m, j \in n \quad (45)$$

$$x_{ij} = 1 \Rightarrow s_i + P_{ij} + Q_{ij} + I_{ij} \leq t_j + E_j, \forall i \in m, j \in n \quad (46)$$

FullQuery Budget Constraints (47) ensure the cost of query i executing on j should be within its budget, B_i , for full query where the original dataset is fully processed satisfying SLAs. C_{ij} is the resource cost to process i on j .

$$C_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (47)$$

SampleQuery Budget Constraints (48) ensure if the original large dataset of query cannot be fully processed that query accuracy of i has to be traded-off due to tight budgets and deadlines, where SC_{ij} is the cost of data sampling and query execution time to process i on sampled dataset on resource j . This should be less than the query budget B_i .

$$SC_{ij} * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (48)$$

SplitQuery Budget Constraints (49) ensure if the full dataset has to be split for parallel processing due to tight deadline requirements, the budget B_i should be sufficient to cover query cost as the sum of PC_{ij} as the cost of splitting data; QC_{ij} as the cost of processing i on the split datasets, and IC_{ij} as the cost of integrating all processing results of the split data to generate the final results using resource j .

$$(PC_{ij} + QC_{ij} + IC_{ij}) * x_{ij} \leq B_i, \forall i \in m, j \in n \quad (49)$$

Budget-Based Available Resource Constraints (50) ensure that if the resource cost to utilize j is greater than the budget specified by i including FullQuery, SampleQuery, and SplitQuery, resource j should be excluded from the available selection pool of resources to support the optimization solver to find a timely scheduling solution in a smaller search space and hence improve the performance of the optimization algorithm.

$$x_{ij} = 0, \text{ if cost of } j \text{ violate budget of } i, \forall i \in m; j \in n \quad (50)$$

Accuracy-Based Available Resource Constraints (51) eliminate j that does not have the required capacity to satisfy the accuracy requirement of at least one query from the selectable resource space to support the optimization solver to obtain the optimization scheduling solution in the reduced search space with better time-efficiency. **FullQuery Accuracy Constraints** ensure that for FullQuery, full accuracy is guaranteed so that resources with sufficient capacity to execute the complete datasets are inputted to the optimization solver to select the best resource configurations for the optimal scheduling solution. **SampleQuery Accuracy Constraints** ensure the cloud resource configurations input to the solver satisfy the accuracy requirements of queries including full accuracy and reduced accuracy to provides reliable data analytics solutions. The accuracy requirements of SampleQueries determine the required sample sizes and associated cloud resources to execute the data samples. Higher query accuracies require larger sample sizes and more computing resources to process the queries. **SplitQuery Accuracy Constraints** input only resources with sufficient capacity that guarantees full accuracy query processing on split datasets to the ILP solution space to support the optimization solver to efficiently select the best configurations for the optimized scheduling solution.

$$x_{ij} = 0, \text{ if } j \text{ violate accuracy of } i, \forall i \in m; j \in n \quad (51)$$

7.3 Resource Scheduling

The profit optimization resource scheduling algorithms are based on the ILP formulation of the proposed scheduling problem to tackle big data analytics challenges in cloud environments. The proposed resource scheduling solutions enable the following three big data analytics scenarios. **Scenario 1:** queries with sufficient deadlines and budgets to process the full datasets are admitted and processed. **Scenarios 2:** queries with sufficient budgets but tight deadlines, where data splitting can be applied to split large datasets into smaller split datasets for parallel processing for significantly reduced times to tackle the deadline challenges. Data splitting incurs higher resource costs due to additional computation required for data splitting and integration operations and hence sufficient budgets are required. **Scenario 3:** queries with limited budgets and tight deadlines, where data sampling is utilized to obtain data samples for quicker query processing with reduced costs. If any query not admitted, the rejection reasons of queries are output to users to enable modification of request specifications to resubmit to the AaaS platform for re-admission.

7.3.1 Splitting and Sampling (SS) Based Profit Optimization Algorithm

Successfully admitted queries are sent to the resource scheduler for resource provisioning and query execution. For each BDAA, the optimization scheduler obtains *fullQueries* that process original full datasets, *splitQueries* that process split small datasets, and *sampleQueries* that process data samples. It combines all the queries as *anaQueries* to process all the data analytics requests. The resource scheduler gets all available resource configurations as available resources *availRes*. It applies heuristic-based scheduling algorithm to generate a near-optimal resource configuration, *heuRes*, as the resources to input to the optimization scheduler. This can significantly reduce the problem space of the optimization that enable the scheduler to generate the scheduling solutions significantly faster than search the whole space with all resource configurations. Afterwards, the resource scheduler utilizes *SSScheduling* method to generate optimization resource scheduling solution. If the optimization solver cannot generate resource scheduling solution within the specified scheduling timeout, heuristic scheduling algorithm is applied to generate near-optimal scheduling solution to enable the AaaS platform to deliver SLA-guaranteed AaaS. (Lines 1-11).

The resource scheduler applies *autoScaling* method (Lines 12-14) to apply the scheduling solution by automatic resource provision. The resource scheduler interprets the scheduling solutions by applying *queryAssignment* method to assign queries to resources in the given execution sequences. Finally, the scheduler provision resources to execute the queries according to the resource scheduling solution.

Method *SSScheduling* (Lines 17-22) applies the modeling and formulation of the optimization resource scheduling problem with overall optimization Objective Z subject to constraints (1-51). *SSScheduling* first obtains the updated input information of queries, resources, and the applications in the AaaS platform. Then it defines the optimization scheduling objective as Z and constraints as (1-51). It further defines the timeout as the latest time for the optimization solver to delivery scheduling solution in the AaaS platform. Finally, *SSScheduling* is applied to solve the mixed ILP scheduling problem. If the optimization solver is able to return scheduling solution before the timeout, a feasible optimization scheduling solution is generated and to-be-applied. Otherwise, the heuristic is applied to generate scheduling solution to satisfy SLAs.

Method *autoScaling* (Lines 24-26) elastically scales resources up by creating new resources when existing resources cannot provision sufficient capacity for query execution with SLA guarantees. Meanwhile, it scales resources down when the platform is over-provisioning resources by terminating idle resources for cost-saving at platform defined periodic checkpoints.

Method *resHeuristic* (Lines 27-31) first gets available created resources in the AaaS platform as *avaResources*. It then selects all the resources that are able to run at least one query satisfying requirements of QoS as *exeResources*. Afterwards, the *resHeuristic* method apply heuristics to pre-schedule all the queries to *exeResources* and scale resources up by selecting new resources as *newResources* to execute queries if the resource capacity is under-provisioned. Finally, the *resHeuristic* method combine the *exeResources* and *newResources* as *heuResources* that are used as near-optimal configuration inputs to the optimization scheduling algorithms.

Splitting and Sampling Based Profit Optimization Scheduling Algorithm

Input: existing cloud resources, cloud resource configurations, BDAAAs, big data analytics requests

Output: scheduling solution for cloud resources

```
1: for every BDAA  $\in$  the BDAA set
2:   fullQueries  $\leftarrow$  obtain queries that process full datasets
3:   splitQueries  $\leftarrow$  obtain queries that process split datasets
4:   sampleQueries  $\leftarrow$  obtain queries that process sample datasets
5:   anaQueries  $\leftarrow$  fullQueries + splitQueries + sampleQueries
6:   availRes  $\leftarrow$  get available resource configurations of BDAA
7:   heuRes  $\leftarrow$  resHeuristic (availRes, resConfigs, anaQueries, BDAA)
8:   solution  $\leftarrow$  SSScheduling (anaQueries, heuRes, BDAA)
9:   if there is no feasible solution is produced within scheduling timeout
10:     solution  $\leftarrow$  heuristicScheduling (heuRes, anaQueries, BDAA)
11:   end if
12:   autoScaling (solution)
13:   queryAssignment (solution)
14:   resourceProvision (solution)
15: end for
16: nextScheduling()
17: Procedure: SSScheduling (anaTasks, BDAA, heuResources)
18:   getUpdatedPlatformInfo (anaQueries, heuRes, BDAA)
19:   defineResourceSchedulingObj (Objective Z)
20:   defineResourceSchedulingConstraints (Constraints 1-51)
21:   definePlatformTimeOut (timeout)
22:   solveResourceScheduleProblem()
23:   solution  $\leftarrow$  generateScheduleSolution ()
24: Procedure: autoScaling (solution)
25:   scaleUpResources (solution) to create new resources
26:   scaleDownResources (solution) to terminate existing resources
27: Procedure: resHeuristic (resources, resConfigs, anaQueries, BDAA)
28:   avaResources  $\leftarrow$  get resources available to execute at least one query
29:   exeResources  $\leftarrow$  select executable resources from avaResources
30:   newResources  $\leftarrow$  use heuristics to determine new resources
31:   heuResources  $\leftarrow$  exeResources + newResources
32: Procedure: queryAssignment (solution)
33:   assign query to a specific PE || resource || a cluster of resources
34:   queryQueue  $\leftarrow$  bind query to the queue in the order of the solution
35: Procedure: heuristicScheduling (heuRes, anaQueries, BDAA)
36:   sort anaQueries based on the maximum delayable time of queries
37:   assign anaQueries to cloud resources with minimal start time
38:   solution  $\leftarrow$  generateScheduleSolution ()
39: Procedure: resourceProvision (solution)
40:   queryExecution (solution)
41:   queryResults (solution)
42:   resourcePeriodicCheck ()
```

Method *queryAssignment* (Lines 32-34) interprets the scheduling solutions by assigning queries to the specified resources that can be a processing element such as a CPU core, a single resource, i.e., VM, and a cluster of VMs. The queries are assigned to the queue in the execution sequences specified by the scheduling solutions for SLA guarantees.

Method *heuristicScheduling* (Lines 35-38) applies maximum delay time based scheduling method that sorts the queries in the order of the available maximum delay time and gives priority to the query with the least delay times. It then schedules queries to the existing resources that can start execution at the minimum estimated start time for the best performances of query execution. Finally, it creates the query scheduling and resource provisioning solutions to elastically provision resources for query execution.

Method *resourceProvision* (Lines 39-42) is used to provision resources for query execution on the specified resources. It further returns the query results after execution. Finally, it utilizes *resourcePeriodicCheck* to scale resources down when the resource capacity is over-provisioned for cost saving purpose while scaling resources up when the resources are under-provisioned for SLA guarantees.

7.3.2 Profit Optimization (PO) Algorithm

PO [11] algorithm also aims to maximize the profits while delivering AaaS solutions with minimized response times applying ILP models. PO supports multi-objective optimization that automatically provisions resources for query execution to minimize resource costs and query times subject to constraints including resource capacity constraints, query budget constraints, query deadline constraints, task execution times constraints, task execution sequence constraints, task dependency constraints, and data locality constraints. PO is limited to schedule queries with sufficient budgets and deadlines. However, PO cannot tackle big data analytics challenges where big data needs to be processed with limited budgets and tight deadlines to support fast and reliable problem solving and decision making in many application domains including e-commerce and healthcare. In this work, we propose SS that can deliver optimal scheduling solutions to support big data analytics with limited budgets and tight deadlines by utilizes sampling and splitting based scheduling to provide reliable, cost-efficient, and timely AaaS. PO supports scheduling solutions for FullQueries.

7.3.3 SPlitting (SP) Based Profit Optimization Algorithm

To better understand the performance of SS, we compare the SS algorithm to SP, which supports splitting based profit optimization resource scheduling that serves same objectives as SS and PO to optimize the platform profits and minimize queries times for users with SLA guarantees. SS algorithm maximize profits of AaaS platforms subject to constraints (1)-(2), (4)-(8), (10)-(28), (38)-(47), (49)-(51) to support data splitting based resource scheduling. For queries with tight deadlines that cannot be admitted by PO, if sufficient budgets are given, data splitting is used to split big data into smaller datasets for parallel processing that reduces the query responses to tackle tight deadline challenges. SP supports scheduling solutions for FullQueries and SplitQueries.

7.3.4 SAmpling (SA) Based Profit Optimization Algorithm

We compare the performance of SS to SA that supports data sampling based optimization scheduling for cloud resources in AaaS platforms. SA has the same profit optimization objective as SS, PO, and SP subject to constraints (1)-(3), (5)-(9), (12)-(37), (47)-(48), (50)-(51). SA is able to admit queries with tight budgets and deadlines where query accuracies can be traded-off to obtain quicker responses and reduced resource costs while accuracy guarantees can be given for reliable decision making by utilizing effective and efficient sampling techniques. SA processes data samples that significantly reduce data processing times and resource costs. SA is able to support resource scheduling for FullQueries and SampleQueries.

7.4 Performance Evaluation

We conduct extensive experiments to analyze algorithm performances of SS compared to PO, SP, and SA for admission controls, SLA guarantees, profit maximization, cost minimization, cloud resource configuration, as well as Algorithm Running Time (ART).

7.4.1 Experiment Setup

Experiment Environment: We build the AaaS framework in the Cloudsim [122], a platform for discrete event simulation, which enables on-demand resource provisioning of distributed clouds. This enables the evaluation of algorithm performance by controllable experiments without incurring considerable expensive resources and BDAA costs for big data analytics. IBM CPLEX 5.5 [134] is utilized as the ILP solver. CPLEX provides an statistical toolkit for time-efficient optimization models based on

mathematics modeling and constraint programming. The experiments are designed with different scenarios including real time scheduling, which schedules queries according to real time arbitrary arrivals and periodic scheduling, which schedules queries in periodic Scheduling Intervals (SIs).

Resource Configuration: We simulate four Amazon EC2 datacenters. Each datacenter is composed of 500 nodes. Each node has 400 CPUs, 30TB memory, 10GB/s bandwidth, and 10PB storage. Six types of memory optimized VMs: r4.large, r4.xlarge, r4.2xlarge, r4.4xlarge, r4.8xlarge, and r4.16xlarge [123] are considered. The resources to execute queries vary that include CPU cores, single VMs, or clusters of VMs and are charged by dollars in a pay-as-you-go model.

Data Analytics Workload: We use Big Data Benchmark [124], BlinkDB [152], and ND splitting [57] to generate the data analytics workload. The ND technique splits large datasets to smaller split datasets to generate data splitting workload. The ND workload provides information of resource times and configurations to split large datasets. BlinkDB is utilized to generate the data sampling workload that details the resource time and configurations to sample large datasets and bounded accuracy for reliable decision making. The benchmark provides details of the query response times and cloud resource configurations utilizing different data analytics frameworks with details of the data size, data type, and data location in the Amazon EC2 Cloud. We model the resource times for splitting and sampling based query execution on BDAAs for different resource configurations and scheduling scenarios based on the above data analytics workload.

Query Request Information: *Query submission times* are generated by one-minute Poisson arrival. *Query types* include join, scan, aggregation, and user defined functions. *BDAA*s are Hadoop (BDAA₁), Hive on Spark (without caching) (BDAA₂), and Hive on Spark (with caching) (BDAA₃). *Resource times* are provided by the big data analytics workload. *Users* utilize the AaaS service; *Deadlines* include tight and loose deadlines [6]. *Budgets* include tight and loose budgets [6]. Accuracies with confidence intervals include 80.0%, 85.0%, 90.0%, 95.0%, and 100.0%.

7.4.2 Admission Control and SLA Guarantee

The experiments are conducted for real time scheduling and periodic scheduling with varied SIs from 1 to 6 minutes. Results show that all admitted queries for SS, SP, SA,

and PO are successfully processed satisfying QoS requirements. Results show the effectiveness of the admission control of SS, SP, SA, and PO by supporting 100% guaranteed SLAs. SS admits more queries with increased admission rates of [18%, 24%], [9%, 12%], [27%, 36%] compared to SP, SA, and PO accordingly, as shown in Figure 7.1. The results show the significant performance advantages of SS to enable higher admission rates to enlarge market share, increase user satisfaction, and generate higher profits.

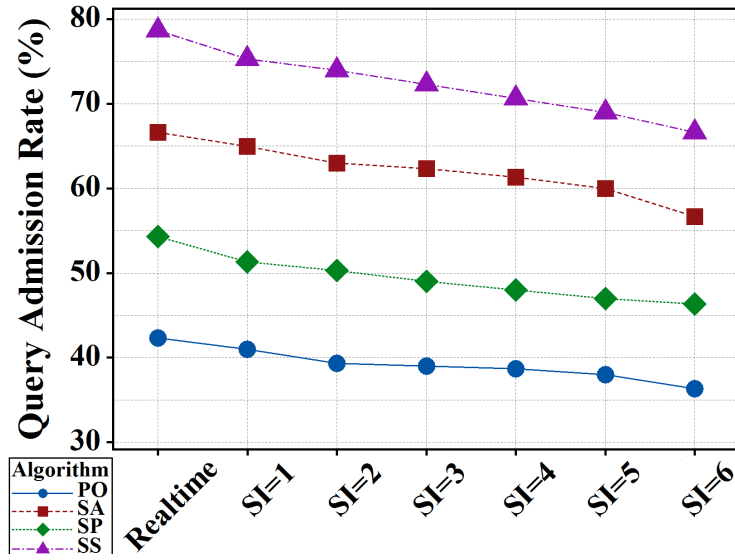


Figure 7.1: Query Admission Rates.

7.4.3 Profit Maximization

The profit enhancement advantages of SS are evaluated for the entire query workload with results shown in Figure 7.2. SS creates significantly higher profits, which is [47%, 58%], [10%, 16%], [82%, 92%] higher than SP, SA, and PO accordingly for all scheduling scenarios. The enhanced profits stem from the higher query admissions and optimized resource utilization with better configurations and query scheduling solutions when higher number of queries are scheduled by the optimization solver. There is an overall profit decreasing trend for all scheduling algorithms as queries with tighter deadlines are rejected for longer SIs. It is not applicable to apply real time scheduling as significant resource consumptions incur due to frequent scheduling. This also decreases the performance advantages of optimization algorithms. Thus, SI=1 is selected as the most suitable SI to admit more queries as well as create higher profits.

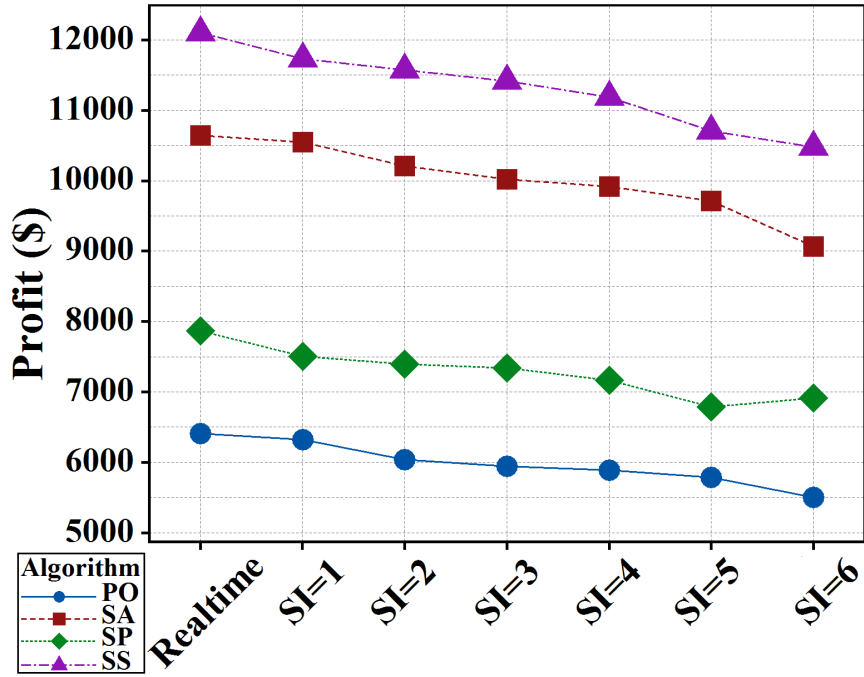


Figure 7.2: Profits.

We analyze the data processing methods for SS, SP, SA, and PO. SS supports FullQueries, SplitQueries, and SampleQueries, while SA supports FullQueries and SampleQueries, SP supports SplitQueries and FullQueries, while PO only support FullQueries. SS, SP, SA processes [82%, 88%], [24%, 28%], [56%, 60%] higher number of queries than PO accordingly, as shown in Figure 7.3. PO processes the least queries due to limited budgets and tight deadlines for full processing of big data. SS is able to process significant higher number of queries as data sampling and splitting based processing methods enable more queries with tight deadlines and limited budgets to be successfully processed, which is followed by SA to exchange accuracy for reduced times and costs, and then followed by SP to speed up big data analytics by parallel processing.

We further conduct performance evaluation of algorithm performances for average query profits. Figure 7.4 shows SA creates [2%, 4%] higher average profits than SS; SS creates [5%, 8%] higher profits than SP; SP creates [1%, 5%] average profits higher than PO. The reason is that SP generates higher query costs as data splitting and integration consume additional resources and hence increase the overall query costs and lead to less query profits; SA incurs significant less resource costs due to approximate

processing of data samples; while SS applies a combination of data splitting and sampling method and hence creates the profits in between of SA and SP. SP outperforms PO in creating higher query profits due to better optimization solutions for larger query admissions and better resource utilizations.

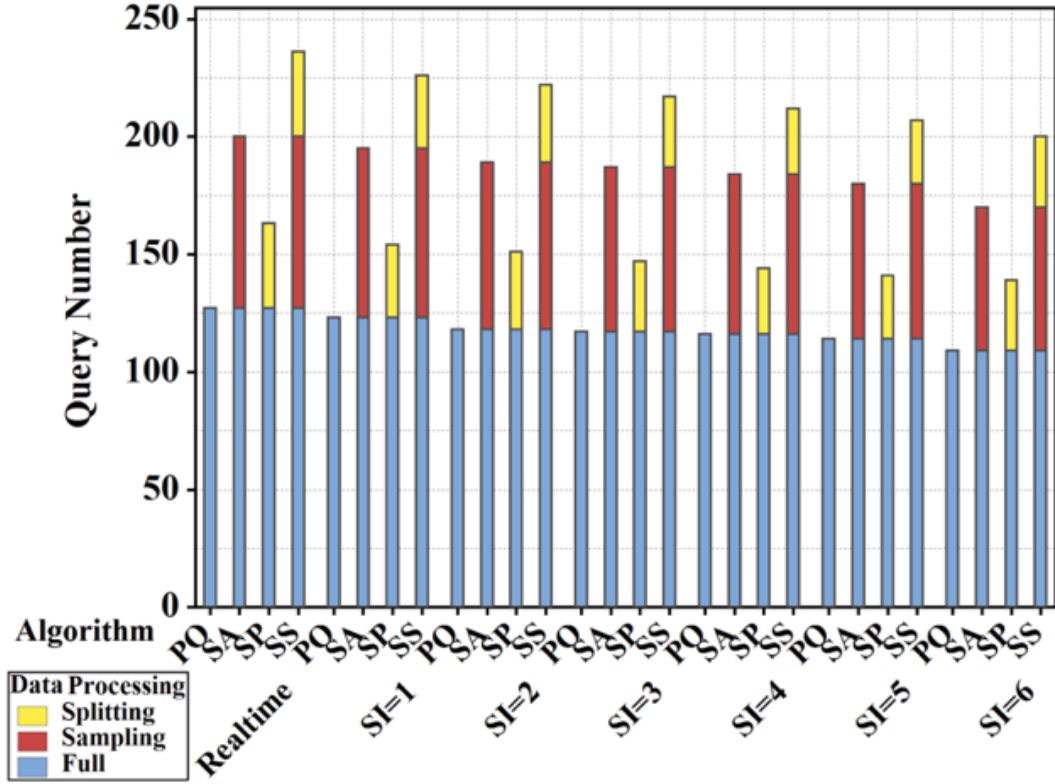


Figure 7.3: Data Processing Methods of Queries.

7.4.4 Cost Saving

We further analyze cost saving advantages of SS. SA and PO have similar resource costs while SS and SP have similar resource costs. SS and SP consume significant higher resources compared to SA and PO, as shown in Figure 7.5. The reason is that for FullQueries, SA and PO requires same computing capacity; for SampleQueries, SA applies data sampling to process additional queries, which consumes significant less resources, and hence the resource costs of SA and PO are similar. For SS and SP, the queries that need full and splitting based processing to process FullQueries and SplitQueries require same computing capacity. Since additional SampleQueries processed by SS requires significant less resources, hence the costs of SS and SP are similar. SS and SP consume significant higher amount of resources due to additional

computing requirements for the splitting and integration operations for SplitQueries; and hence the resource costs are significantly higher than PO and SA. PO consumes significant less resources for the minimum query admissions. SA consumes significant less resources for larger query admissions for SampleQueries. SA is suitable for queries with tight deadlines and budgets whose accuracies can be traded-off, SP is suitable for full queries with tight deadlines and sufficient budgets. SS applies both data sampling and splitting based processing that is suitable for all three type of queries including FullQueries, SplitQueries and FullQueries that create significant performance advantages compared to SA, SS, and PO to support big data analytics under tight budgets and deadlines.

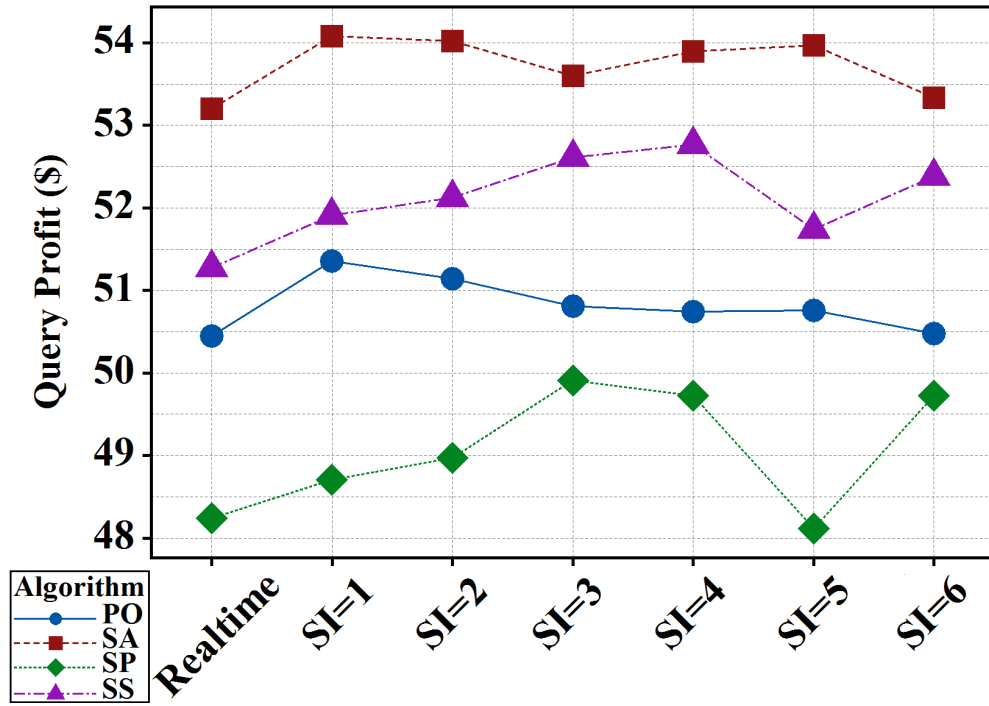


Figure 7.4: Query Profits.

7.4.5 Resource Configuration

The evaluation of resource configurations is conducted for real time and periodic scheduling with results shown in Figure 7.6. Results show that SS utilizes the largest resource configurations for all scheduling scenarios, followed by SP, SA, and PO. The number of configured VMs for SS is [139, 327], for SP is [135, 208], for SA is [41, 53], and for PO is [35, 47]. There is a significant increase of resource configurations for both

SS and SP due to the parallel processing requirements for timely AaaS delivery by scaling up hundreds of VMs to process the queries under tight deadlines. We also noticed that SS and SP utilize a significant large number of r4.16xlarge clusters due to the tight deadline requirements of queries that needs timely parallel processing. R4.16xlarge can offer higher processing capacity, therefore large clusters of r4.16xlarge are set up for tight deadlines. This is consistent with query time minimization objective and SLA guarantee purpose. SS requires highest numbers of resources, which is [2%, 46%] higher compared to SP, 3 to 6 times higher compared to SA, and 4 to 8 times higher compared to PO. The resource configurations show the efficiency of the SS to setup large-scale computing environments for big data processing.

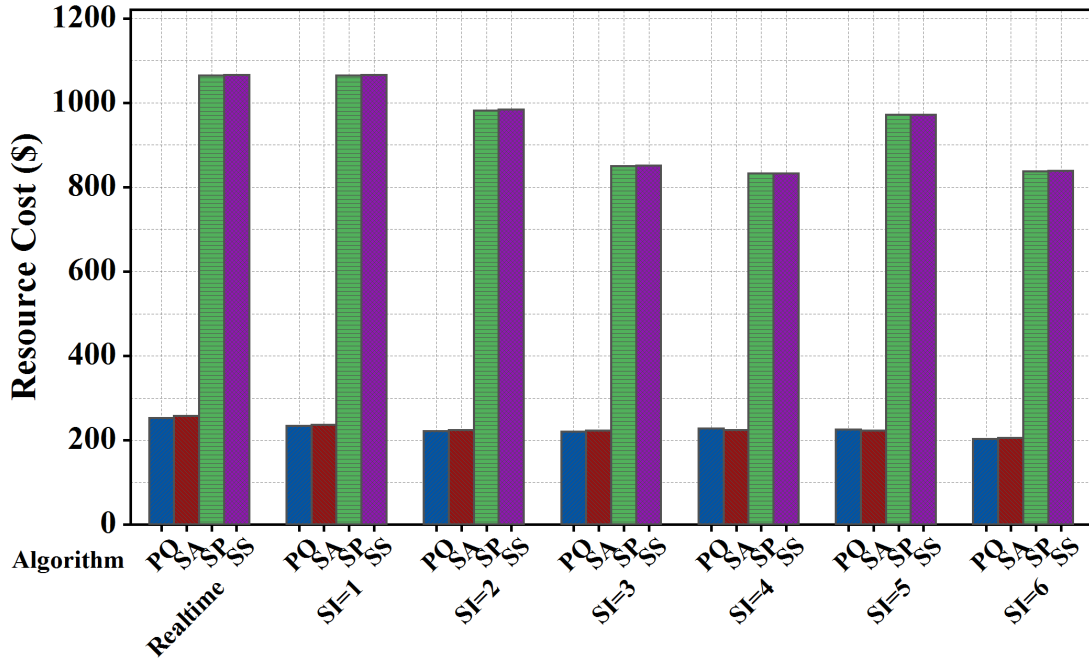


Figure 7.5: Resource Costs.

The average query resource costs for SS, SP, SA, and PO are shown in Figure 7.7. Results show that SP algorithm has the highest average query resource costs in the range of [5.8, 7.0], which is 1.5 times of the resource cost of SS and 3-4 times of resource cost of PO. The average resource cost per query for SA is the lowest. SA significantly saves resource costs and times by applying sampling based resource scheduling that is under the trade-off data accuracy and hence has the limitations of applicability. SP algorithm helps deadline constrained queries where full accuracies are required for big data

processing under the condition that the budget of the query is not a limiting factor. SS applies both data splitting and sampling based scheduling that enables fully processing of big data for deadline constrained scenarios as well as sampling based processing for deadline and budget constrained scenarios. SS is the most suitable algorithm to face big data processing challenges under various QoS requirements to provide optimal solutions for different application scenarios.

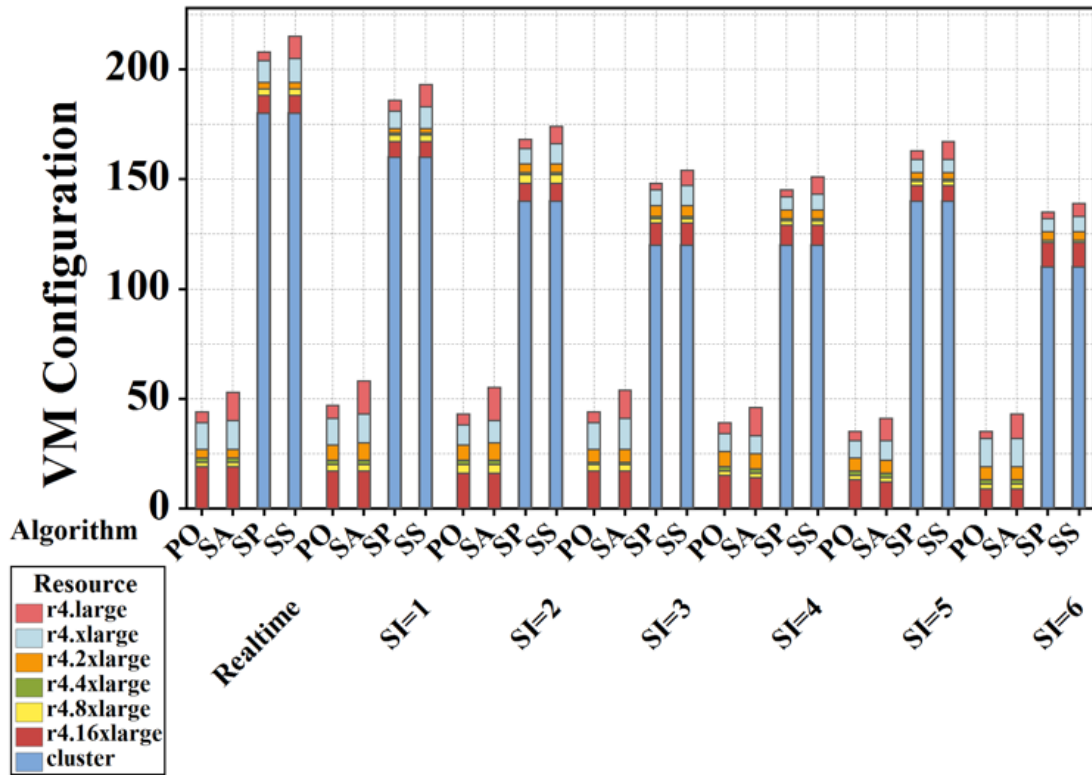


Figure 7.6: Resource Configurations.

7.4.6 Algorithm Performance for BDAAs

The above evaluations are based on results of processing the entire query workloads of three BDAA. We conduct detailed algorithm performance evaluations for every BDAA with scheduling in real time and in periodic with SI=1. The results show that SS outperforms SP, SA, and PO for every BDAA for both real time and periodic scheduling.

The results in Figure 7.8 show that SS can admit the most queries for BDAA with a higher acceptance rate of [36%, 59%], [9%, 24%], and [56%, 104%] compared to SP, SA, and PO accordingly for resource scheduling in real time and in SI=1. Obtained

results show that SS supports higher query admission and profit enhancement with splitting and sampling based scheduling for all BDAAAs. SA has the second largest query admission rates as it is able to accept queries with limited budgets and deadlines by applying sampling based scheduling to process queries on data samples. SP has a [13%, 37%] higher admission rate compared to PO as SP applies splitting based query admission and resource scheduling to admit queries with tight budgets and deadlines and provision resources for parallel processing of split datasets.

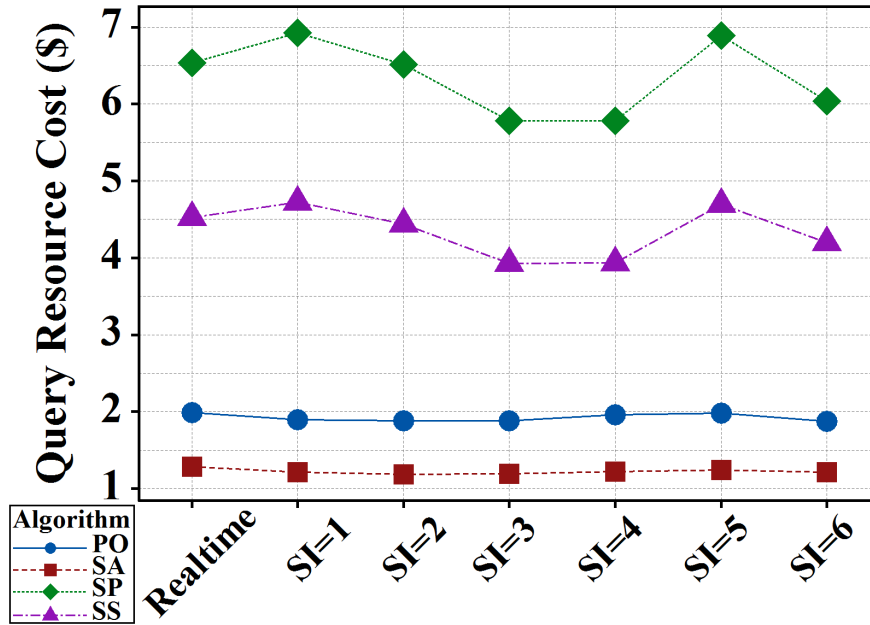


Figure 7.7: Query Resource Costs.

We further analyze the resource configurations of SS, SP, SA, and PO for each BDAAAs, as shown in Figure 7.9. We can see that SS and SP algorithms consume significant larger amount of resources than SA and PO. Clusters of r4.16xlarge VMs are utilized for BDAAAs with the VM number in the range of [50, 80]. The reason is due to higher query workloads that consume increased computing resources for parallel data processing under tight deadlines.

Through further analytics of resource costs per BDAA shown in Figure 7.10, we can see that the resource costs of SS are 3-9 times that of the resource costs of PO for the three BDAAAs, which is consistent with the larger resource requirements for big data processing for the increased number of accepted queries with tight deadlines. The results of resource configurations show the SS support large scale big data computing for data

analytics requests with tight budgets and deadlines. The results show the cost saving advantages of the SS scheduling algorithm that is able to process [36%, 59%] higher number of queries compared to SP but at similar resource costs.

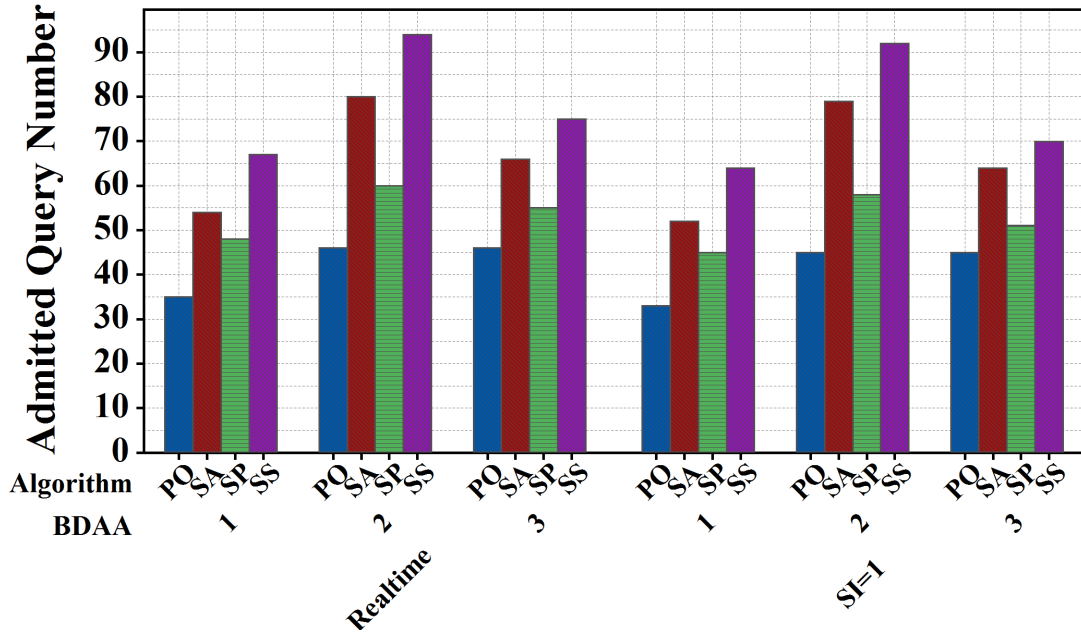


Figure 7.8: Admitted Query Number for BDAAs.

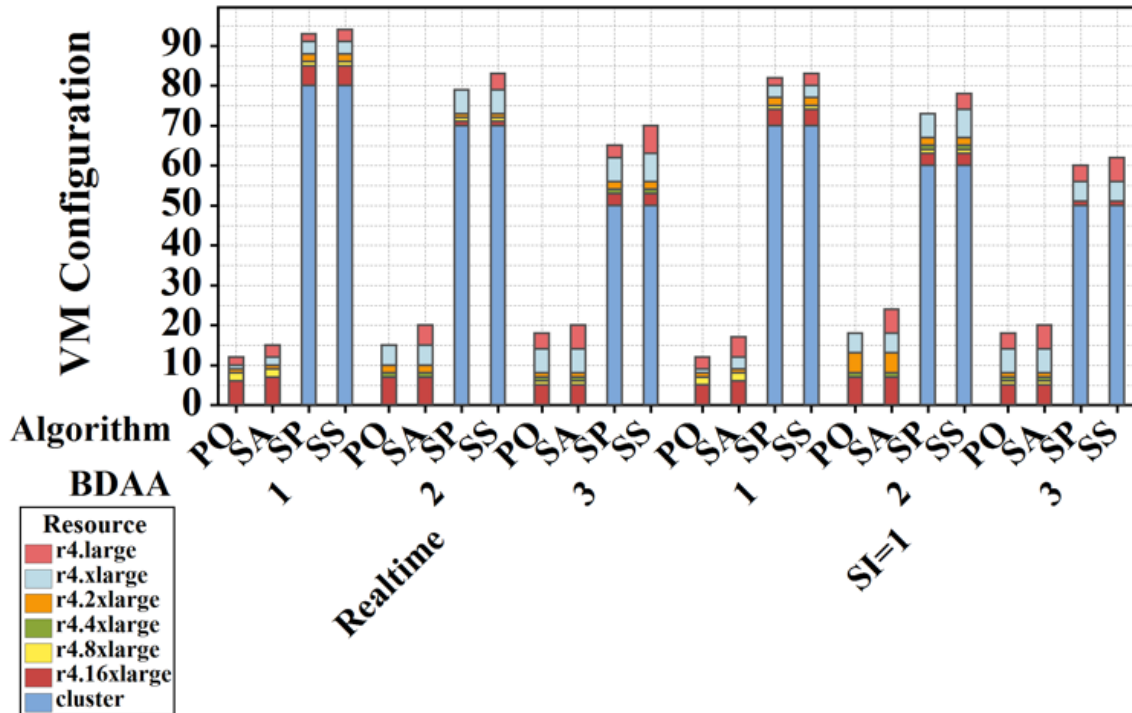


Figure 7.9: Resource Configurations of BDAAs.

Results also show the performance advantages of SS to maximize the platform profits, as shown in Figure 7.11. SS is able to create significant profits of [41%, 72%] compared to SP. Moreover, SS is able to create higher profits compared to SA and PO, which is increased by [7%, 14%] and [13%, 37%]. We can also see that SA has higher cost saving advantages compared to SS and PO. Results show that SA has higher admission rates of [13%, 36%] and cost saving of [28%, 54%] for BDAA 1-3 compared to SS. For similar resource configurations and costs, SS is able to admit [42%, 76%] higher queries and create [46%, 88%] higher profits than PO. The performance of cost saving advantages of SA comes from sampling based data processing that is at the sacrifice of accuracy to reduce resource costs and query times. SP and PO are able to provide fully accurate data processing for reliable problem solving and decision making that supports the application scenarios where fully processing the original large datasets is required.

7.4.7 Algorithm Running Time (ART) Analysis

We analyze the ART for the algorithm performance evaluation of SS. The average ART of SS, SP, SA, and PO algorithms for the complete query workload with real time scheduling and periodic scheduling. Obtained results show an increasing trend of ART in Figure 7.12 as more queries, resources, constraints with larger SIs formulate more complex optimization scheduling problems for all algorithms. Results show SS, SP, SA, and PO can deliver optimal resource scheduling solutions within 5-9 milliseconds. This shows the time efficiency of all optimization algorithms in delivering optimal solutions. Results show that the ART does not limit the algorithm performance of SS.

Mixed ILP optimization problem with constraints programming can take considerable time when problem space is very large. To deliver timely solutions with SLA guarantees, backup heuristic solutions are required if optimization algorithms cannot deliver scheduling solutions within SIs when problem space is very large for unexpectedly large query arrivals. We have designed a range of heuristics as backup algorithms for unexpectedly large number of online query arrivals that might exceed the maximum times allowed for optimization algorithms to deliver solutions satisfying deadline requirements. In the experiments, the timeout value is set to the 90% of the SI that enables sufficient times for the heuristics to deliver results within 10% of the SI.

The aim is to tackle unexpected large query arrivals that exceed the capacity of the optimization algorithms to delivery scheduling solutions within SIs to guarantee SLAs.

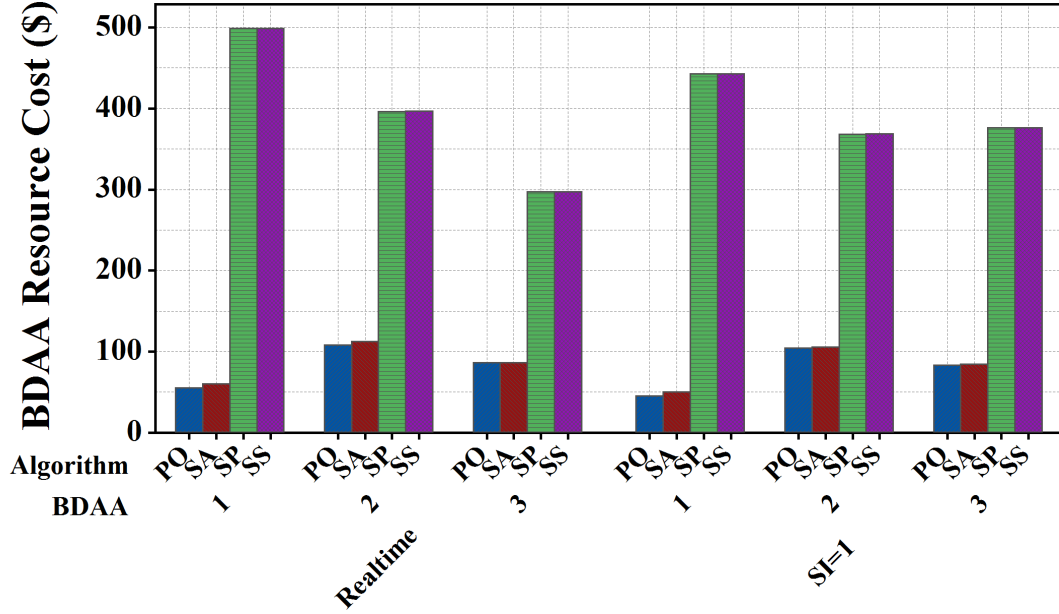


Figure 7.10: Resource Costs of BDAA.

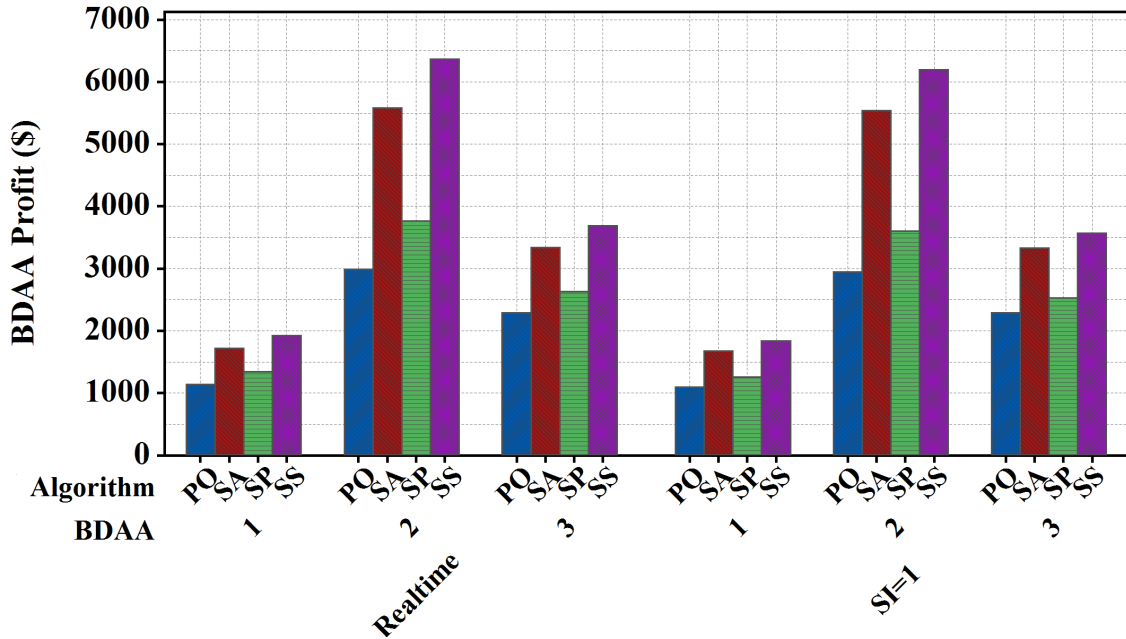


Figure 7.11: Profits of BDAA.

If the optimization algorithms are not able to deliver scheduling solutions within given timeout, heuristics are automatically applied to deliver scheduling solutions. The

proposed heuristics also generate near-optimal cloud resource configurations to input to the optimization algorithms to avoid a thorough search for all available resource configurations in the search space to achieve higher algorithm efficiency and timely scheduling solutions. We keep improving the performance of the heuristics to provide better near-optimal resource inputs and serve as better backup algorithms.

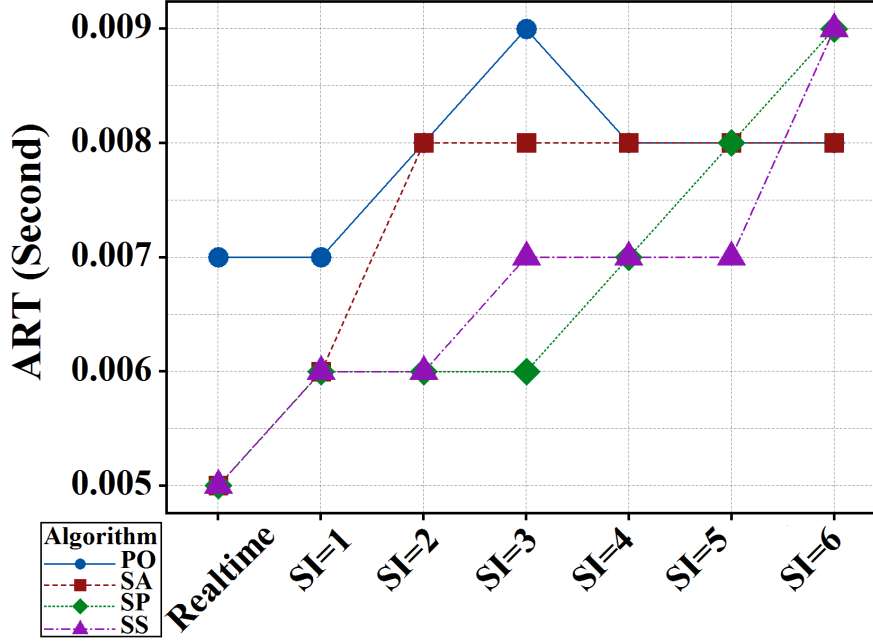


Figure 7.12: Algorithm Running Time.

In summary, SS significantly outperforms SP, SA, and PO that serves as a comprehensive and adaptive scheduling algorithm, which is suitable for both sampling based approximate data processing and splitting based parallel data processing targeting different big data analytics scenarios. All of the results show that SS significantly outperforms SP, SA, and PO under numerous performance indicators including enhanced profits, reduced cloud resource costs, increased query admissions with efficient, automatic, and large-scale configurations for the whole query workloads, for single BDAAAs, and for both real time and periodic scheduling. SS significantly increases the profits of the AaaS platform due to maximized request admission rates and optimized scheduling solutions. SS is able to support big data computing with optimized resource management solutions that tackle tight deadline and budget constraints of big data analytics and enables timely, cost-effective, and reliable AaaS delivery.

7.5 Related Work

We compare our research work to the most related works that solve cloud resource scheduling problems for big data analytics with the support of different data processing and computing technologies.

Zhao et al. [6] propose resource management algorithms to enhance profits for AaaS platforms where the solutions may be sub-optimal. Zhao et al. [11] support a profit optimized scheduling algorithm to optimize the profits for AaaS platforms with minimized response times to deliver optimal solutions with SLA guarantees on budgets and deadlines. The works process queries while budgets and deadlines are sufficient for big data analytics. Zhao et al. [13] provide data-splitting based profit maximization scheduling solutions, which split original large dataset to process smaller split datasets as representatives of original large dataset that trade-off query accuracies for reduced costs and less times and deliver AaaS with SLA guarantees on budgets and deadlines. However, this work cannot deliver AaaS solutions with accuracy guarantees.

Zhao et al. [22] provide a data sampling-based profit optimization scheduling solution that trade-off accuracy for quicker times and reduced costs. The solution is able to provide reliable AaaS with SLA guarantees on budgets, deadlines, and accuracies. However, this solution cannot process queries that require fully accurate query processing while deadlines are limiting factors. Zhao et al. [7] provide data splitting based profit optimization algorithms that split large datasets into smaller datasets and parallel process split datasets to speed up queries while guaranteeing SLAs on budgets and deadlines. However, this solution does not support big data processing where budgets are limiting factors.

There are a number of works focus on resource scheduling for MapReduce tasks. Xiao et al. [95] propose a cost-aware approach for big data processing over distributed datacenters to minimize costs for MapReduce applications satisfying budget constraints while they do not consider QoS requirements on deadlines and accuracies or guarantee SLAs. Lin et al. [91] propose a multi-objective selective algorithm to predict job execution duration and space occupation for MapReduce tasks while they do not consider QoS requirements of tasks or SLA guarantees. Malekimajd et al. [85] formulate an optimization model for admission control and resource allocation of MapReduce jobs to minimize resource costs with deadline guarantees. They do not consider SLA guarantees on QoS parameters of budgets or accuracies or support automatic scheduling.

There are a number of works focus on workflow scheduling. Arabnejad et al. [113] support budget and deadline aware resource scheduling algorithms for scientific workflows, however, they do not tackle big data processing challenges under tight budgets and deadlines or guarantee SLAs. Lin et al. [89] study an online resource scheduling strategy for scientific workflows on hybrid clouds for deadline constrained applications at lower price. They do not consider accuracy or budget requirements or tackle big data challenges of tight budgets and deadlines. Zhang et al. [112] focus on deadline constrained workflow scheduling to maximize resource utilizations and success rates to meet task deadlines. They do not consider admission control to guarantee SLAs, budget or accuracy requirements, or support automatic scheduling.

Gianniti et al. [81] propose a tool for cloud configurations with minimum cost deployment for big data platforms satisfying QoS constraints of tasks based on nonlinear programming. However, they do not consider SLA guarantees or tackle big data challenges of tight deadlines or limited budgets. Dou et al. [104] propose a cloud resource co-allocation method for load balance scheduling of HPC applications in big data systems. However, they do not consider QoS parameters, cost optimization, or automatic scheduling.

Our work aims to offer general big data AaaS to various domains that can support various application models with different task structures and dependencies. Moreover, none of the above related works is able to provide automatic and scalable multi-objective resource management solutions to optimize profits for AaaS platforms and minimize query times while guaranteeing SLAs on budgets, deadlines, and accuracies. We apply data sampling and splitting based resource scheduling algorithms to deliver cost-effective, timely, and reliable AaaS to tackle big data challenges on tight budgets and deadlines, where comes the novelty of our work.

7.6 Conclusion and Future Works

We propose automatic and efficient optimization resource scheduling algorithms that support data splitting and sampling based big data processing, which tackle challenges of big data analytics subject to tight query budgets and deadlines. The proposed multi-objective optimization algorithms optimize the profits for AaaS platforms while offering SLA guaranteed AaaS with minimized query times. Extensive experiments are conducted to evaluate the algorithm performance of SS for the entire query workload of

all BDAAAs using both realtime and periodic scheduling. Experimental evaluations show the automaticity and efficiency of SS that outperforms SA, SP, and PO. SS significantly increases query admissions, enhances profits, reduces resource costs, provides large-scale resource configurations while delivering AaaS with SLA guarantees for the entire query workload, for all BDAAAs, and for both real time scheduling and periodic scheduling. Our proposed algorithms provide optimal scheduling solutions to tackle big data analytics challenges under tight deadlines and limited budgets to support cost-effective, fast, and reliable problem solving and decision making for various application domains. In the future, we will continue working on proposing optimization algorithms to support automatic and efficient resource scheduling in the AaaS platforms and provide optimization scheduling solutions to tackle big data analytics challenges in cloud computing environments.

8 Conclusions and Future Directions

8.1 Conclusions

The value that can be extracted from big data demands users and organizations to utilize data analytics technologies to gain insights to make better decisions and address problems across a wide range of domains. Cloud computing greatly benefits big data analytics by offering on-demand and scalable computing infrastructures, tailored platforms, and offering applications as services that can reduce analytical costs and times by elastically provisioning resources based on user requirements. Big data analytics requires large-scale computing that can consume significant amounts of resources that may incur high costs if too many resources are used for an extended time period. AaaS platforms provision AaaS as consumable services to users in a pay-as-you-go model.

To support AaaS platforms, it is necessary to provide optimal resource management solutions for cloud providers to deliver AaaS with SLA guarantees to enhance profits, increase market share and improve their robustness and hence their reputation. To tackle the dynamism of query requests, heterogeneity of cloud resources, support different QoS requirements, tackle diverse data processing times and avoid expensive resource costs, it is necessary to support efficient and automatic admission control and resource scheduling algorithms to maximize profits and minimize query times while guaranteeing SLAs on budgets, deadlines, and/or make compromises with regards to the accuracy of data analytics results.

In this thesis, we proposed a range of profit optimization resource management solutions that address these issues through realistic scenarios.

- **Chapter 3** proposed resource management algorithms for **Problem Scenario 1** to enhance the profits for cloud providers by offering cost-effective, SLA-guarantee based resource scheduling solutions that maximize resource utilization of cloud resources. The proposed solutions are based on two phase scheduling that maximize resource utilization of existing resources (Stage 1) and subsequently minimize resource costs to scale up new resources to execute queries (Stage 2). The algorithms are based on auto-scaling of dynamic resource requirements for requests and minimizing resource consumption to avoid over-

provisioning and/or or under-provisioning of resources to save costs. This work supported query processing on complete (large) datasets where budgets and deadlines were met by the computational capacity of the cloud with SLA guarantees used to support controllable data processing times and costs.

- **Chapter 4** provided resource management algorithms for **Problem Scenario 2** with optimal solutions that enhanced the solutions in Chapter 3 where results were sub-optimal due to the two-phase scheduling mechanism. The proposed algorithms offered profit-maximization and time-minimization solutions with support for heuristic algorithms that offered near-optimal configurations to reduce the problem space and enable optimization algorithms to deliver solutions more rapidly. The heuristics also served as backup algorithms for unexpectedly large query arrivals where optimization algorithms might cause bottlenecks due to the large search space required to prevent SLA violations. For query processing with deadline and budget constraints where accuracy might be traded-off, representative datasets were processed to further enhance profits by increasing query admissions and reducing data processing times and costs, whilst supporting SLAs on budgets and deadlines.
- **Chapter 5** proposed data sampling-based admission control and associated cloud resource scheduling algorithms for **Problem Scenario 3** to support big data analytics solutions under deadline and budget constraints. If sufficient budgets and deadlines were given, queries could be processed with full accuracy. If tight deadlines and/or limited budgets were given where processing the entire dataset would violate constraints, data sampling could be used to process smaller sampled datasets to tackle time and cost challenges with approximate accuracy guarantees. The proposed resource management solutions significantly benefit big data analytics under budget, deadline, and accuracy constraints by supporting fully accurate query processing as well as sampling-based query processing for timely, cost-efficient, and reliable AaaS.
- **Chapter 6** explored SLA-aware and data splitting-based resource management solutions for **Problem Scenario 4** to enable big data analytics with reliable and

rapid results as might be required for application domains such as banking and emergency service situations. The proposed algorithms split large datasets into smaller datasets and support parallel processing of the split datasets to speed up query processing. The proposed algorithms tackle deadline constrained big data processing scenarios, where accurate results are required and SLAs need to be maintained to support reliable and fast decision making. The scheduling algorithms enabled flexible task dependencies and structures, supporting auto-scaling and data locality-based scheduling, prioritized resource utilization, dynamic reaction to varying resource demands, data processing considerations of locality, optimized query times, and minimized resource costs with support for optimized resource configurations.

- **Chapter 7** considered automatic and scalable multi-objective resource management algorithms for big data analytics with heterogeneous QoS requirements on budgets, deadlines, and accuracies. The proposed algorithms optimized profits for AaaS platforms and minimized query times while guaranteeing SLAs that address **Problem Scenario 5**. For queries within deadlines and budgets, data analytics solutions could be delivered under budget and deadline constraints with full accuracy by processing the entire large datasets for the most accurate results. For queries with sufficient budgets but limited deadlines where fast results were required, data splitting could be applied for parallel processing on split datasets to reduce the data processing times to address the time constraints. For queries with limited budgets and deadlines and accuracy requirements, data sampling-based methods could be applied to tackle time-limited and cost-limited big data processing scenarios. This work prioritizes query processing with full result accuracy and trades-off query accuracy only when necessary to get the most reliable results. The proposed solutions provide a comprehensive mechanism to enhance profits by maximizing query admission and optimizing resource provisioning with elastic and large-scale resource configurations to deliver cost-effective, fast, and reliable AaaS with SLA guarantees on budgets, deadlines, and result accuracies.

8.2 Future Directions

There are several open questions that might be studied in the future to advance this work and address limitations of this thesis.

- **Question 1:** *How to tackle big data processing challenges to satisfy heterogeneous functional and non-functional requirements of data analytics requests?*
 - This remains an open question that to be further explored. In this thesis, we studied data splitting and sampling techniques to reduce data processing times and/or costs by parallel and approximate processing. The applicability of data processing techniques depends on various factors including requirements of data analytics requests, BDAAAs, and data characteristics. For example, in Chapter 5, sampling was applied to reduce data processing times and resource costs as query accuracies could be traded-off. However, in Chapter 6, data sampling could not be applied as full accuracy was required by queries and BDAAAs, hence data splitting was applied to speed up query processing. Advanced data processing technologies to support heterogeneous BDAAAs and data analytics requests can be developed to deliver AaaS with faster response times, reduced resource costs, and offer reliable results to benefit various application domains. Appropriate data processing techniques-based resource management solutions need to be further explored based on functional and non-functional requirements of different data analytics requests and BDAAAs.
- **Question 2:** *How to deliver reliable, timely, and cost-efficient AaaS without compromising the quality of AaaS?*
 - This question needs to be further addressed, e.g. through efficient solutions for resource management with SLA enforcement. In this thesis, we studied profit optimization solutions with budget, deadline, and accuracy guaranteed AaaS delivery. SLA guarantees on budgets and deadlines have been widely studied, while SLA guarantees on accuracy

requirements need to be further explored. In Chapter 5 and Chapter 7, accuracy guarantees to deliver reliable results were limited to BDAAAs where accuracy-bounded query processing was supported. Further research needs to be conducted to tackle heterogeneous problem scenarios and support various data analytics requests. Maximizing profits and minimizing query times while guaranteeing accuracy requirements for heterogeneous queries and BDAAAs needs to be further explored, e.g. through SLA-based resource management algorithms. It is important to ensure SLAs accommodate and support data analytics requests from various application domains to continuously improve the quality of AaaS for better user experiences.

- **Question 3:** *How to provide SLA-guaranteed resource management solutions that satisfy heterogeneous QoS requirements of queries?*
 - This question needs to be addressed as different BDAAAs and queries may have different QoS requirements based on user demands. Comprehensive resource management algorithms that enable heterogeneous QoS-based admission control and resource scheduling may provide solutions with higher applicability, flexibility, and usability. In this thesis, QoS requirements of budget, deadline and/or accuracy were studied for queries however data security and consistency issues were not considered. Other QoS factors such as security, availability, consistency and privacy may be further explored based on the requirements of different application domains. Examples of these might be healthcare and commerce. SLA-aware resource management algorithms to support heterogeneous QoS requirements might also be explored as extensions to the work in this thesis.
- **Question 4:** *How to provide accurate BDAA profiling to support reliable and automatic admission control and resource scheduling solutions?*
 - This question needs to be addressed to provide reliable input for AaaS platforms to admit queries and schedule resources and enable diverse AaaS delivery. BDAA profiling information is essential to estimate

query times and costs to make admission and scheduling decisions. Obtaining application profiles requires expert knowledge. In this thesis, BDAA profiles are assumed to be reliably provisioned by BDAA providers. BDAA profiling-enabled resource management solutions might be further developed as extension to this work. Future works on application profiling related to the performance of algorithms may be beneficial in dealing with dynamic features of data analytics requests on heterogeneous cloud resources, whilst enabling SLA guarantees on query processing to provide profit optimization solutions with better resource provisioning.

- **Question 5:** *How to provide fault-tolerant resource management solutions for AaaS platforms to optimize profits and guarantee SLAs?*
 - The proposed resource management solutions in this thesis were formulated and tested on simulated cloud environments where system and resource bottlenecks and failures were not considered. In cloud environments, resource failures and system bottleneck can be unavoidable. How to react to unexpected under-performance and failures of resources and platforms remains a challenging question. Moreover, unexpected task failures can be caused by various issues such as failure of disks and loss of data from intermediate tasks that would affect query execution of dependent tasks. Such kind of faults/failures that would lead to SLA violations and expensive penalty costs should be prevented by effective fault-tolerant mechanisms. Fault-tolerant solutions are essential to help AaaS platforms recover from unexpected faults to enable reliable AaaS delivery, ensure SLAs for higher user satisfaction, enhance profits and avoid expensive penalty costs being incurred as a result of SLA violations. Fault-tolerant resource management solutions might be further explored and how they cope with unexpected failures whilst guaranteeing SLAs.
- **Question 6:** *How to discover and select suitable and efficient BDAAAs, AaaS solutions, and infrastructure resources to support diverse data analytics requests from various application domains?*

- This question needs to be continuously explored and supported through the development and advances in cloud service discovery and selection technologies. Discovery and selection of suitable and efficient resources enable AaaS platforms to provide general AaaS to serve various application domains and satisfy various functional and non-functional needs related to data analytics requests. Moreover, resource management solutions need to support dynamic discovery and selection of new BDAAAs, specific AaaS solutions, and cost-effective resource configurations to accept more data analytics requests, optimize profits, increase market share, and improve the quality of AaaS delivery and hence reputation. With advances in data processing technologies and improved performance and capacity of cloud resources, diverse data analytics services, higher-quality service delivery might be further explored.

Bibliography

- [1] L. Cai and Y. Zhu, “The challenges of data quality and data quality assessment in the big data era,” *Data Science Journal*, vol. 14, no. 2, pp. 1-10, 2015.
- [2] X. Jin, B. W. Wah, X. Cheng and Y. Wang, “Significance and challenges of big data research,” *Big Data Research*, vol. 2, no. 2, pp. 59–64, 2015.
- [3] S. Lavalley, E. Lesser, R. Shockley, M. S. Hopkins and N. Kruschwitz, “Big data, analytics and the path from insights to value,” *MIT Sloan Management Review*, vol. 52, no. 2, pp. 21–32, 2011.
- [4] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto and R. Buyya, “Big data computing and clouds: Trends and future directions,” *Journal of Parallel and Distributed Computing*, vol. 79, pp. 3-15, 2015.
- [5] K. Kambatla, G. Kollias, V. Kumar and A. Grama, “Trends in big data analytics,” *Journal of Parallel and Distributed Computing*, vol. 74, pp. 2561-2573, 2014.
- [6] Y. Zhao, R. N. Calheiros, G. Gange, K. Ramamohanarao and R. Buyya, “SLA-based resource scheduling for big data analytics as a service in cloud computing environments,” in *Proceedings of IEEE International Conference on Parallel Processing*, pp. 510–519, 2015.
- [7] Y. Zhao, R.N. Calheiros, A.V. Vasilakos, J. Bailey and R.O. Sinnott, “Profit maximization and time minimization admission control and resource scheduling for cloud-based big data analytics-as-a-service platforms,” in *Proceedings of International Conference of Web Services*, pp. 26-47, 2019.
- [8] L. Qian, Z. Luo, Y. Du and L. Guo, “Cloud computing: An overview,” in *IEEE International Conference on Cloud Computing*, pp. 626-631, 2009.
- [9] P. Melville, V. Sindhvani and R. Lawrence, “Social media analytics: Channeling the power of the blogosphere for marketing insight,” in *Proceedings of WIN*, pp.1-5, 2009.
- [10] D. Serrano, S. Bouchenak, Y. Kouki, F. A. Oliveira, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes and P. Sens, “SLA guarantees for cloud services,” *Future Generation Computer Systems*, vol. 54, pp. 233-246, 2016.
- [11] Y. Zhao, R. N. Calheiros, J. Bailey and R. O. Sinnott, “SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in

- cloud computing environments,” in Proceedings of IEEE International Conference on Big Data, pp. 432-441, 2016.
- [12] X. Sun, B. Gao, L. Fan and W. An, “A cost-effective approach to delivering analytics as a service,” in Proceedings of IEEE 19th International Conference on Web Services, pp. 512-519, 2012.
- [13] Y. Zhao, R. N. Calheiros, G. Gange, J. Bailey and R. O. Sinnott, “SLA-based profit optimization for resource scheduling of big data analytics-as-a-service in cloud computing environments,” IEEE Transactions on Cloud Computing, pp. 1-18, 2018.
- [14] R. Mian, P. Martin, A. Brown, M. Zhang, S. Fiore and G. Aliosio, “Managing data intensive workloads in the cloud,” Grid and Cloud Database Management, pp. 235–260, 2011.
- [15] N. A. B. Mary and K. Jayapriya, “An extensive survey on QoS in cloud computing,” International Journal of Computer Science and Information Technologies, vol. 5, no. 1, pp. 1-5, 2014.
- [16] Y. Zhao, A.V. Vasilakos, J. Bailey and R. O. Sinnott, “Profit optimization for splitting and sampling based resource management in big data analytics-as-a-service platforms in cloud computing environments,” in Proceedings of International Conference on eScience, pp. 156-167, 2019.
- [17] M. Chen, S. Mao and Y. Liu, “Big data: A survey,” Mobile Networks and Applications, vol. 19, no. 2, pp. 171–209, 2014.
- [18] G. Bell, T. Hey and A. Szalay, “Beyond the data deluge,” Science, vol. 323, no. 5919, pp. 1297–1298, 2009.
- [19] C. P. Chen and C. Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on big data,” Information Sciences, no. 275, pp. 314–347, 2014.
- [20] R. Kapdoskar, S. Gaonkar, N. Shelar, A. Surve and S. Gavhane, “Big data analytics,” vol. 4, no. 10, pp. 518–520, 2015.
- [21] F. Zulkernine, P. Martin, Y. Zou, M. Bauer, F. Gwadry-Sridhar and A. Aboulmaga, “Towards cloud-based analytics-as-a-service (CLAaaS) for big data analytics in the cloud,” in Proceedings of 2013 IEEE International Congress on in Big Data, pp. 62-69, 2013.
- [22] Y. Zhao, R. N. Calheiros, A. V. Vasilakos, J. Bailey and R. O. Sinnott, “SLA-aware and deadline constrained profit optimization for cloud resource management in big

- data analytics-as-a-service platforms,” in Proceedings of IEEE International Conference on Cloud Computing, pp. 146-155, 2019.
- [23] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility,” *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009.
- [24] W. Voorsluys, J. Broberg and R. Buyya, “Introduction to cloud computing,” *Cloud Computing: Principles and paradigms*, pp. 3-37, 2011.
- [25] P. Mell and T. Grance, “The NIST definition of cloud computing,” National Institute of Standards and Technology, pp. 1-7, 2011.
- [26] Y. Xing and Y. Zhan, “Virtualization and cloud computing,” in Proceedings of Future Wireless Networks and Information Systems, pp. 305-312, 2012.
- [27] L. Malhotra, D. Agarwal and A. Jaiswal, “Virtualization in cloud computing,” *Journal of Network and Computer Applications*, vol. 4, no. 2, pp.1-3, 2014.
- [28] S. Dustdar, Y. Guo, B. Satzger and H. L. Truong, “Principles of elastic processes,” *IEEE Internet Computing*, vol. 15, no. 5, 66-71, 2011.
- [29] I. Breskovic, M. Maurer, V. C. Emeakaroha, I. Brandic and S. Dustdar, “Cost-efficient utilization of public SLA templates in autonomic cloud markets,” in Proceedings of IEEE International Conference on Utility and Cloud Computing, 2011.
- [30] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra and B. Hu, “Everything as a service (XaaS) on the cloud: origins, current and future trends,” in IEEE International Conference on Cloud Computing, pp. 621-628, 2015.
- [31] Y. Duan, Y. Cao and X. Sun, “Various “aaS” of everything as a service,” in Proceedings of IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, pp. 1-6, 2015.
- [32] Q. Chen, M. Hsu and H. Zeller, “Experience in continuous analytics as a service (CaaS),” in Proceedings of International Conference on Extending Database Technology, pp. 509-514, 2011.
- [33] R. S. Barga, J. Ekanayake and W. Lu, “Project Daytona: Data analytics as a cloud service,” in Proceedings of IEEE International Conference on Data Engineering, pp. 1317-1320, 2012.

- [34] O. Runsewe and N. Samaan, "Cloud resource scaling for time-bounded and unbounded big data streaming applications," *IEEE Transactions on Cloud Computing*, pp. 1–14, 2018.
- [35] F. Chen, C. Zhang, F. Wang and J. Liu, "Crowdsourced live streaming over the cloud," in *Proceedings of IEEE Infocom*, pp. 2524–2532, 2015.
- [36] M. Kim, S. Han, Y. Cui, H. Lee, H. Cho and S. Hwang, "CloudDMSS: Robust hadoop-based multimedia streaming service architecture for a cloud computing environment," *Cluster Computing*, vol. 17, no. 3, pp. 605–628, 2014.
- [37] H. Li, K. Lu and S. Meng, "BigProvision: A provisioning framework for big data analytics," *IEEE Network*, vol. 29, pp. 50–56, 2015.
- [38] R. Yang, X. Ouyang, Y. Chen, P. Townend and J. Xu, "Intelligent resource scheduling at scale: a machine learning perspective," in *Proceedings of IEEE Symposium on Service-Oriented System Engineering*, pp. 132–141, 2018.
- [39] M. Ahmad, A. Aboulnga, S. Babu and K. Munagala, "Interaction-aware scheduling of report-generation workloads," *VLDB Journal*, vol. 20, no. 4, pp. 589–615, 2011.
- [40] S. Yousfi, M. V. University-agdal, D. Chiadmi and M.V. University-agdal, "Big data-as-a-service solution for building graph social networks," in *Proceedings of IEEE International Conference on Cloud Technologies and Applications*, pp. 1–6, 2015.
- [41] A. G. Garcia, I. Blanquer Espert and V. Hernández García, "SLA-driven dynamic cloud resource management," *Future Generation Computer Systems*, vol. 31, pp. 1–11, 2014.
- [42] C. Curino, E. P. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan and N. Zeldovich. "Relational cloud: A database-as-a-service for the cloud," in *Proceedings of Biennial Conference on Innovative Data Systems Research*, pp. 235–241, 2011.
- [43] R. A. Popa, C.M. Redfield, N. Zeldovich and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proceedings of ACM Symposium on Operating Systems Principles*, pp. 85–100, 2011.
- [44] V. H. Nguyen, T. K. Dang, N. T. Son and J. Kung, "Query assurance verification for dynamic outsourced XML databases," in *Proceedings of Conference on Availability, Reliability and Security*, pp. 689–696, 2007.

- [45] V. Narasayya, S. Das, M. Syamala, B. Chandramouli and S. Chaudhuri, "SQLVM: Performance isolation in multi-tenant relational database-as-a-service," 6th Biennial Conference on Innovative Data Systems Research, pp. 1-9, 2013.
- [46] P. Wong, Z. He and E. Lo, "Parallel analytics as a service," in Proceedings of International Conference on Management of Data, pp. 25-36, 2013.
- [47] R. Ramakrishnan, B. Sridharan, J. R. Douceur, P. Kasturi, B. Krishnamachari-Sampath, K. Krishnamoorthy, P. Li, M. Manu, S. Michaylov, R. Ramos and N. Sharman, "Azure data lake store: a hyperscale distributed file service for big data analytics," in Proceedings of ACM International Conference on Management of Data, pp. 51-63, 2017.
- [48] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107-113, 2008.
- [49] S. Sakr, A. Liu and A. G. Fayoumi, "The family of MapReduce and large-scale data processing systems", ACM Computing Surveys, vol. 46, no. 1, pp.1-44, 2013.
- [50] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman and P. Maechling, "Scientific workflow applications on Amazon EC2," in Proceedings of IEEE International Conference on e-Science Workshops, pp. 59-66, 2009.
- [51] L. Singh and S. Singh, "A survey of workflow scheduling algorithms and research issues," International Journal of Computer Applications, vol. 74, no. 15, 2013.
- [52] T. Lynn, P. Rosati, A. Lejeune and V. Emeakaroha, "A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms," in Proceedings of IEEE International Conference on Cloud Computing Technology and Science, pp. 162-169, 2017.
- [53] H. Khazaei, C. McGregor, J. M. Eklund and K. El-Khatib, "Real-time and retrospective health-analytics-as-a-service: A novel framework," JMIR Medical Informatics, vol. 3, no. 4, pp.1-20 2015.
- [54] P. Kumar, S. Kumar and S. Wu, "SLA based healthcare big data analysis and computing in cloud network", Journal of Parallel and Distributed Computing, vol. 119, pp. 121–135, 2018.
- [55] A. Jindal, A. Dua, N. Kumar, A. K. Das, A. V. Vasilakos and J. J. P. C. Rodrigues, "Providing healthcare-as-a-service using fuzzy rule based big data analytics in cloud computing," IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 5, pp. 1605–1618, 2018.

- [56] F. Tordini, M. Aldinucci, P. Viviani, I. Merelli and P. Lio, “Scientific workflows on clouds with heterogeneous and preemptible instances,” in Proceedings of International Conference on Parallel Computing, pp. 605-614, 2017.
- [57] H. Zhang, Y. Zhao, C. Pang and J. He, “Splitting large medical data sets based on normal distribution in cloud environment,” IEEE Transactions on Cloud Computing, pp. 1-14, 2015.
- [58] S. Agarwal, A. P. Iyer, A. Panda, S. Madden, B. Mozafari and I. Stoica, “Blink and it’s done: Interactive queries on very large data,” in Proceedings of VLDB Endowment, pp. 1902-1905, 2012.
- [59] Y. Hua, B. Xiao and X. Liu, “NEST: Locality-aware approximate query service for cloud computing,” in Proceedings of IEEE Infocom, pp. 1303–1311, 2013.
- [60] M. Cardoso, C. Wang, A. Nangia, A. Chandra and J. Weissman, “Exploring MapReduce efficiency with highly-distributed data,” in Proceedings of International Workshop on MapReduce and Its Applications, pp. 27-34, 2011.
- [61] S. Fernandes and J. Bernardino, “What is BigQuery?” in Proceedings of International Database Engineering & Applications Symposium, pp. 202-203, 2015.
- [62] L. Petrazickis, M. Butuc and B. Steinfeld, “Crunching big data with Hadoop and BigInsights in the cloud,” in Proceedings of Conference of the Center for Advanced Studies on Collaborative Research, pp. 241-242, 2012.
- [63] K. Shvachko, H. Kuang, S. Radia and R. Chansler, “The Hadoop distributed file system,” in Proceedings of IEEE Symposium on Mass Storage Systems and Technologies, pp. 1–10, 2010.
- [64] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth and B. Saha, “Apache Hadoop YARN: Yet another resource negotiator,” in Proceedings of Annual Symposium on Cloud Computing, pp. 1-16, 2013.
- [65] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker and I. Stoica, “Shark: SQL and rich analytics at scale,” in Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 13-24, 2013.
- [66] M. Zaharia, M. Chowdhury, T. Das, A. Dave and J. Ma, “Fast and interactive analytics over Hadoop data with Spark,” USENIX Login, vol. 37, no. 4, pp. 45-51, 2012.

- [67] A. Thusoo, J. Sarma, N. Jain and Z. Shao, "Hive: A warehousing solution over a map-reduce framework," in Proceedings of VLDB Endowment, vol. 2, no. 2, pp. 1626-1629, 2009.
- [68] T. Liu, M. Martonosi, T. Liu and M. Martonosi, "Impala: A middleware system for managing autonomic, parallel sensor systems ting," in Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 107-118, 2003.
- [69] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy and C. Curino, "Apache Tez: A unifying framework for modeling and building data processing applications," in Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 1357-1369, 2015.
- [70] Y. Wen, J. Liu, W. Dou, X. Xu, B. Cao and J. Chen, "Scheduling workflows with privacy protection constraints for big data applications on cloud," Future Generation Computer Systems, vol. 108, pp. 1084-1091, 2020.
- [71] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," Journal of Network and Computer Applications, vol. 114, pp. 108-122, 2018.
- [72] H. Kllapi, E. Sitaridi, M. M. Tsangaris and Y. Ioannidis, "Schedule optimization for data processing flows on the cloud," in Proceedings of International Conference on Management of Data, pp. 289-300, 2011.
- [73] [Online]. Analytics as a Service Market. Available: <http://www.alliedmarketresearch.com/analytics-as-a-service-market>
- [74] R. Mian, P. Martin and J. Vazquez-Poletti, "Provisioning data analytic workloads in a cloud," Future Generation Computer Systems, vol. 29, no. 6, pp. 1452-1458, 2013.
- [75] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar and R. Buyya, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," Journal of Network and Computer Applications, vol. 45, pp. 108-120, 2014.
- [76] L. Gu, D. Zeng, P. Li and S. Guo, "Cost minimization for big data processing in geo-distributed data centers," IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 3, pp. 314-323, 2014.
- [77] D. Zeng, L. Gu and S. Guo, "Cost minimization for big data processing in geo-distributed data centers," Cloud Networking for Big Data, pp. 59-78, 2015.

- [78] M. C. Calzarossa, M. L. Della and D. Tessler, "A methodological framework for cloud resource provisioning and scheduling of data parallel applications under uncertainty," *Future Generation Computer Systems*, vol. 93, pp. 212–223, 2019.
- [79] Q. Xia, Z. Xu, W. Liang and A. Y. Zomaya, "Collaboration- and fairness-aware big data management in distributed clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 1941-1953, 2015.
- [80] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee and A.Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," in *Proceedings of International Symposium on High Performance Distributed Computing*, pp. 229-238, 2011.
- [81] E. Gianniti, M. Ciavotta and D. Ardagna, "Optimizing quality-aware big data applications in the cloud," *IEEE Transactions on Cloud Computing*, pp. 1-16, 2018.
- [82] J. Liu, J. Ren, W. Dai, D. Zhang and P. Zhou, "Online multi-workflow scheduling under uncertain task execution time in IaaS clouds," *IEEE Transactions on Cloud Computing*, vol. 7161, no. c, 2019.
- [83] Q. Shao, S. Liu, L. Pan, C. Yang and T. Niu, "A market-oriented heuristic algorithm for scheduling parallel applications in big data service platform," in *Proceedings of IEEE International Conference on Computer Software & Applications*, pp. 677–686, 2018.
- [84] C. Li, J. Tang and Y. Luo, "Hybrid cloud adaptive scheduling strategy for heterogeneous workloads," *Journal of Grid Computing*, vol. 17, no. 3, pp. 419–446, 2019.
- [85] M. Malekimajd, D. Ardagna, M. Ciavotta, E. Gianniti, M. Passacantando and A. M. Rizzi, "An optimization framework for the capacity allocation and admission control of MapReduce jobs in cloud systems," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 5314-5348, 2018.
- [86] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-12, 2011.
- [87] M. Ebrahimi, A. Mohan and S. Lu, "Scheduling big data workflows in the cloud under deadline constraints," in *Proceedings of IEEE International Conference on Big Data Computing Service and Applications*, pp. 33-40, 2018.

- [88] Y. Wang, Y. Xia and S. Chen, "Using integer programming for workflow scheduling in the cloud," in *Proceedings of IEEE International Conference on Cloud Computing*, pp. 138-146, 2017.
- [89] B. Lin, W. Guo and X. Lin, "Online optimization scheduling for scientific workflows with deadline constraint on hybrid clouds," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 11, pp. 3079-3095, 2016.
- [90] W. Zheng, Y. Qin, E. Buringo, D. Zhang and J. Chen, "Cost optimization for deadline-aware scheduling of big-data processing jobs on clouds," *Future Generation Computer Systems*, pp. 244-255, 2018.
- [91] B. Lin, W. Guo, N. Xiong, G. Chen, A.V. Vasilakos and H. Zhang, "A pretreatment workflow scheduling approach for big data applications in multicloud environments," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp.581-594, 2016.
- [92] F. Jrad, J. Tao, I. Brandic and A. Streit, "SLA enactment for large-scale healthcare workflows on multi-cloud," *Future Generation Computer Systems*, vol. 43, pp. 135–148, 2015.
- [93] M. Alrokayan, A. Vahid Dastjerdi and R. Buyya, "SLA-aware provisioning and scheduling of cloud resources for big data analytics," in *Proceedings of IEEE International Conference on Cloud Computing in Emerging Markets*, pp. 1-8, 2014.
- [94] M. Mattess, R. N. Calheiros and R. Buyya, "Scaling MapReduce applications across hybrid clouds to meet soft deadlines," in *Proceedings of 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 629-636, 2013.
- [95] W. Xiao, W. Bao, X. Zhu and L. Liu, "Cost-aware big data processing across geo-distributed datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp.3114-3127, 2017.
- [96] F. Zhang, J. Cao, K. Li, S. U. Khan and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," *Future Generation Computer Systems*, vol. 37, pp. 309–320, 2014.
- [97] H. Cao and C. Q. Wu, "Performance optimization of budget-constrained MapReduce workflows in multi-clouds," in *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 243-25, 2018.

- [98] Z. Tang, L. Jiang, J. Zhou, K. Li and K. Li, "A self-adaptive scheduling algorithm for reduce start time," *Future Generation Computer Systems*, vol. 43–44, no. 0, pp. 51–60, 2015.
- [99] R. Kune, P. K. Konugurthi, A. Agarwal, R. R. Chillarige and R. Buyya, "Genetic algorithm based data-aware group scheduling for big data clouds," in *Proceedings of IEEE/ACM International Symposium on Big Data Computing*, pp. 96–104, 2014.
- [100] Y. Sun, F. Lin and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II," *Wireless Personal Communications*, vol. 102, no. 2, pp. 1369–1385, 2018.
- [101] K. Fan, Y. Ren and Z. Yan, "A dynamic resource allocation method for load-balance scheduling over big data platforms," pp. 1349–1354, 2018.
- [102] H. Tang, C. Li, J. Bai, J. H. Tang and Y. Luo, "Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud–edge environment," *Computer Communications*, vol. 134, no. May 2018, pp. 70–82, 2019.
- [103] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang and I. Raicu, "Optimizing load balancing and data-locality with data-aware scheduling," in *Proceedings of IEEE International Conference on Big Data*, pp. 119–128, 2015.
- [104] W. Dou, X. Xu, X. Liu, L. T. Yang and Y. Wen, "A resource co-allocation method for load-balance scheduling over big data platforms," *Future Generation Computer Systems*, vol. 86, pp. 1064–1075, 2018.
- [105] K. Kaur, N. Kumar, S. Garg and J.J. Rodrigues, "EnLoc : Data locality-aware energy-efficient scheduling scheme for cloud data centers," in *Proceedings of IEEE International Conference on Communications*, pp. 1–6, 2018.
- [106] Y. Shao, C. Li, J. Gu, J. Zhang and Y. Luo, "Efficient jobs scheduling approach for big data applications," *Computers & Industrial Engineering*, vol. 117, pp. 249–261, 2018.
- [107] A. Sfrent and F. Pop, "Asymptotic scheduling for many task computing in big data platforms," *Information Sciences*, vol. 319, pp. 71–91, 2015.
- [108] M. Malawski, G. Juve, E. Deelman and J. Nabrzyski, "Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS cloud," *Future Generation Computer Systems*, vol. 48, pp.1–18, 2015.

- [109] W. Dai, L. Qiu, A. Wu and M. Qiu, "Cloud infrastructure resource allocation for big data applications," *IEEE Transactions on Big Data*, vol. 4, no. 3, pp. 313 - 324, 2018.
- [110] A. C. Zhou, B. He, X. Cheng and C. T. Lau, "A declarative optimization engine for resource provisioning of scientific workflows in IaaS clouds," in *Proceedings of International Symposium High Performance Parallel Distributed Computing*, pp. 223–234, 2015.
- [111] J. C. Anjos, K. J. Matteussi, P. R. D. Souza, A. S. Veith, G. Fedak, J. L. V. Barbosa and C. R. Geyer, "Enabling strategies for big data analytics in hybrid infrastructures," in *Proceedings International Conference on High Performance Computing and Simulation*, pp. 869–876, 2018.
- [112] D. Zhang, W. Yan, E. Bugingo, W. Zheng and J. Chen, "A benchmark approach and its toolkit for online scheduling of multiple deadline-constrained workflows in big-data processing systems," *Future Generation Computer Systems*, vol. 85, pp. 222-234, 2018.
- [113] V. Arabnejad, K. Bubendorfer and B. Ng, "Budget and deadline aware e-science workflow scheduling in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 29-44, 2018.
- [114] M. Kollenstart, E. Harmsma, E. Langius, V. Andrikopoulos and A. Lazovik, "Adaptive provisioning of heterogeneous cloud resources for big data processing," *Big Data and Cognitive Computing*, vol. 2, no. 3, pp.15-32, 2018.
- [115] F. Zhang, J. Cao, W. Tan, S.U. Khan, K. Li and A.Y. Zomaya, "Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp.338-351, 2014.
- [116] Q. Xiang, X. T. Wang, J. J. Zhang, H. Newman, Y. R. Yang and Y. J. Liu, "Unicorn: Unified resource orchestration for multi-domain, geo-distributed data analytics," *Future Generation Computer Systems*, vol. 93, pp. 188–197, 2019.
- [117] [Online]. IBM Social Media Analytics. Available: <http://www-03.ibm.com/software/products/en/social-media-analytics-saas>
- [118] M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere and O. Vincent, "Experiments in mixed-integer linear programming," *Mathematical Programming*, vol. 1, pp. 76-94, 1971.

- [119] S. K. Garg, S. K. Gopalaiyengar and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in Proceedings of International Conference on Algorithms and Architectures for Parallel Processing , pp. 371-384, 2011.
- [120] R. Benayoun, J. De Montgolfier, J. Tergny and O. Laritchev, "Linear programming with multiple objective functions: Step method (STEM)," Mathematical Programming, vol. 1, pp. 366-375, 1971.
- [121] H. Isermann, "Linear lexicographic optimization," Operations-Research-Spektrum, vol. 4, pp. 223-228, 1982.
- [122] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011.
- [123] [Online]. Amazon EC2 Instance Types. Available: <http://aws.amazon.com/ec2/instance-types/>
- [124] [Online]. Big Data Benchmark. Available: <https://amplab.cs.berkeley.edu/benchmark/>
- [125] J. Schad, J. Dittrich and J. A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing and reducing variance," in Proceedings of VLDB Endowment, vol. 3, no. 1-2, pp. 460-471, 2010.
- [126] R. N. Calheiros and R. Buyya, "Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds," in Proceedings of International Conference on Web Information System Engineering, pp. 171-184, 2012.
- [127] [Online]. LpSolver. Available: <http://lpsolve.sourceforge.net/5.5/>
- [128] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in Proceedings of IEEE International Conference on Cloud Computing, pp. 423-430, 2012.
- [129] I. A. T. Hashem, I. Yaqoob, N. Badrul Anuar, S. Mokhtar, A. Gani and S. Ullah Khan, "The rise of 'big data' on cloud computing: Review and open research issues," Information Systems, vol. 47, pp. 98-115, 2014.
- [130] C. Ji, Y. Li, W. Qiu, U. Awada and K. Li, "Big data processing in cloud computing environments," in Proceedings of International Symposium on Pervasive Systems, Algorithms and Networks, pp. 17-23, 2012.

- [131] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker and I. Stoica, “Spark: Cluster computing with working sets”, in Proceedings of USENIX Conference Hot Topics in Cloud Computing, pp. 1-7, 2010.
- [132] M. K. A. B. V. Bittorf, T. Bobrovitsky, C. C. A. C. J. Erickson, M. G. D. Hecht, M. J. I. J. L. Kuff, D. K. A. Leblang, N. L. I. P. H. Robinson, D. R. S. Rus, J. R. D. T. S. Wanderman and M. M. Yoder, “Impala: A modern, open-source SQL engine for Hadoop,” in Proceedings of Biennial Conference on Innovative Data Systems Research, pp. 1-10, 2015.
- [133] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi and M. Zaharia, “Spark SQL: Relational data processing in spark,” in Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 1383-1394, 2015.
- [134] [Online]. IBM ILOG CPLEX Optimization Studio. Available: <https://www.ibm.com/developerworks/downloads/ws/ilogcplex/>
- [135] T. Bicer, D. Chiu and G. Agrawal, “Time and cost sensitive data-intensive computing on hybrid clouds,” in Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 636–643, 2012.
- [136] [Online]. Amazon Elastic MapReduce. Available: <https://aws.amazon.com/elasticmapreduce/>
- [137] [Online]. Google BigQuery. Available: <https://Cloud.google.com/bigquery/>
- [138] [Online]. IBM Big SQL. Available: <http://www-03.ibm.com/software/products/en/ibm-biginights-for-apache-hadoop/>
- [139] [Online]. Relational Cloud. Available: <http://relationalcloud.com>
- [140] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden and I. Stoica, “BlinkDB: Queries with bounded errors and bounded response times on very large data,” in Proceedings of ACM European Conference on Computer Systems, pp. 29-42, 2013.
- [141] [Online]. Tez. Available: <https://www.cloudera.com/products/open-source/apache-hadoop/apache-tez.html>
- [142] K. Ousterhout, P. Wendell, M. Zaharia and I. Stoica, “Sparrow: Distributed, low latency scheduling,” in Proceedings of ACM Symposium on Operating Systems Principles, pp. 69–84, 2013.

- [143] F. Tordini, "A cloud solution for multi-omics data integration," in Proceedings of IEEE International Conference on Scalable Computing and Communication, pp. 559-566, 2016.
- [144] I. A.T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani and U. Khan, "The rise of 'big data' on cloud computing: review and open research issues," Information Systems, vol. 47, pp. 98–115, 2014.
- [145] G. Das and V. Narasayya, "Optimized stratified sampling for approximate query processing," ACM Transactions on Database Systems, vol. 32, no. 2, pp. 1-50, 2007.
- [146] C. W. Tsai, C. F. Lai, H. C. Chao and A. V. Vasilakos, "Big data analytics: a survey," Journal of Big Data, vol. 2, no. 1, pp. 1–32, 2015.
- [147] H. T. Jongen, K. Meer and E. Triesch, "Integer linear programming," Optimization Theory, Springer Science & Business Media, pp. 257–270, 2007.
- [148] R. N. Calheiros and R. Buyya, "Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds," in Proceedings of International Conference on Web Information System Engineering, pp. 171-184, 2012.
- [149] I. Yaqoob, I. A. T. Hashem, A. Gani, S. Mokhtar, E. Ahmed, N.B. Anuar and A. V. Vasilakos, "Big data: from beginning to future," International Journal of Information Management, vol. 36, no. 6, pp.1231-1247, 2016.
- [150] C. Yang , Q. Huang, Z. Li, K. Liu and F. Hu, "Big data and cloud computing: innovation opportunities and challenges," International Journal of Digital Earth, vol. 10, no. 1, pp.13-53, 2017.
- [151] D.P. Acharjya and K. Ahmed, "A survey on big data analytics: challenges, open research issues and tools," International Journal of Advanced Computer Science and Applications, vol. 7, no. 2, pp.511-518, 2016.
- [152] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari and I. Stoica, "Knowing when you're wrong: building fast and reliable approximate query processing systems," in Proceedings of ACM International Conference on Management of Data, pp. 481-492, 2014.
- [153] Y. Zhang, T. Cao, S. Li, X. Tian, L. Yuan, H. Jia, and A. V. Vasilakos, "Parallel processing systems for big data: A survey," in Proceedings of the IEEE, vol. 104, no. 11, pp. 2114-2136, 2016.
- [154] W. Gan, J. C. W. Lin, H. C. Chao, A. V. Vasilakos, and S.Y. Philip, "Utility-driven data analytics on uncertain data," IEEE Systems Journal, 2020.

- [155] J. Wan, P. Zheng, H. Si, N. N. Xiong, W. Zhang, and A. V. Vasilakos, "An Artificial Intelligence Driven Multi-Feature Extraction Scheme for Big Data Detection," *IEEE Access*, vol. 7, pp. 80122-80132, 2019.
- [156] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," in *Proceedings of the IEEE*, vol. 102, no. 1, pp. 11-31, 2014.
- [157] D. Dahiphale, R. Karve, A.V. Vasilakos, H. Liu, Z. Yu, A. Chhajjer, J. Wang, and C. Wang, "An advanced MapReduce: cloud MapReduce, enhancements and applications," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 101-115, 2014.
- [158] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Transactions on Computers*, vol. 65, no. 4, pp. 1172-1184, 2016.
- [159] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A.V. Vasilakos, "The role of big data analytics in internet of things," *Computer Networks*, pp. 459-471, 2017.
- [160] H. Cai, B. Xu, L. Jiang, and A.V. Vasilakos, "IoT-based big data storage systems in cloud computing: perspectives and challenges," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp.75-87, 2016.
- [161] X. Zeng, S. Garg, M. Barika, A. Y. Zomaya, L. Wang, M. Villari, D. Chen, and R. Ranjan, "SLA management for big data analytical applications in clouds: A taxonomy study," *ACM Computing Survey*, vol. 53, no. 3, pp. 1-40, 2020.
- [162] H. Li, Y. Zhao, S. Fang, "CSL-driven and energy-efficient resource scheduling in cloud data center," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 481–498, 2020.
- [163] M. Lavanya, B. Shanthi, and S. Saravanan, "Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment," *Computer Communications*, vol. 151, pp. 183-195, 2020.
- [164] J. Lee, Y. S. Kwon, F. Farber, M. Muehle, C. Lee, C. Bensberg, J. Y. Lee, A. H. Lee, and W. Lehner, "SAP HANA distributed in-memory database system: Transaction, session, and metadata management," in *Proceedings of IEEE International Conference on Data Engineering*, pp. 1165-1173, 2013.

- [165] H. Chen, J. Wen, W. Pedrycz and G. Wu, “Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds,” *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 131–144, 2020.