MANAGEMENT OF SERVICE LEVEL AGREEMENTS FOR BIG DATA ANALYTICS APPLICATIONS IN CLOUD: A LAYER-BASED STUDY

by

XUEZHI ZENG

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

THE AUSTRALIAN NATIONAL UNIVERSITY

©XUEZHI ZENG OCTOBER 2019

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.



XUEZHI ZENG

29 October 2019

Acknowledgments

Throughout my Ph.D. journey, I have received guidance, inspiration, support and help from many amazing people whom I wish to acknowledge and appreciate.

First and foremost, I would like to express my deep gratitude to my principal supervisor, Prof. Rajiv Ranjan for his consistent support, never-ending concern, and superb guidance. He has rendered me great encouragement and support when I experience academic difficulties and personal issues. The completion of my Ph.D. study would not have been possible without his support and help. I would also like to own my sincere appreciation to my co-supervisor, Dr. Saurabh Garg. He has given me valuable suggestion and numerous support. He has resolved my academic doubts, advised me on how to write scientific papers with great patience, and arranged my visit to the University of Tasmania. I appreciate very much to my chair associate supervisor, Associate Prof. Peter Strazdins for his generous support to my research. I wish to extend my sincere gratitude to Prof. Alistair Rendell, my chair of the panel, for his encouragements, insightful comments, and great support throughout my Ph.D. study.

Many thanks go to the Research School of Computer Science of Australian National University (ANU) for granting me scholarships and travel support, which enabled me to pursue doctoral study and attend international conferences. I appreciate the Commonwealth Scientific and Industrial Research Organization (CSIRO) for offering me the opportunity to work with and learn from many brilliant scientists, especially Dr. Prem Prakash Jayaraman and Dr. Carsten Friedrich. Thanks for their help and support. I also thank the University of Tasmania for offering me the visiting opportunity.

Special thanks to Dr. Sanat Bista who offered me great opportunity as a data scientist working with many amazing professionals at an Australian Government agency during my Ph.D. study.

I also appreciate Janette Rawlinson (former School Manager of Computer Science) for her prompt support to my Ph.D. study and staff members at ANU and CSIRO for their help and friendship, in particular: Dr. Ying-Hsang Liu, Dr. Ali Reza Khoshkbar, Dr. Sara Salem Hamouda, Meisong Wang, and Miranda Zhang. It has been a great pleasure and a privilege to work with you all.

Last, but certainly not the least, I wish to acknowledge the efforts of my family. To my beloved wife Yuan, I can never thank you enough for your understanding, support, concern, and sacrifices made for me. I would never have been the person that I am today without you. Thanks my sweet boy Felix for his bringing me joyfulness, courage and driving force in my life and shaping me to be a good father. I would also appreciate my parents, parents-in-law, sisters and other family members who have rendered me enormous love, tolerance, and forgiveness.

Abstract

Nowadays, deployment and operation of big data analytics applications (BDAAs) in cloud are increasingly becoming a trending practice. These applications offer organizations the capabilities of constructing valuable information and extracting actionable insight for enhancing the evidence-based decision-making process. Many leading providers such as Google and Amazon provision such analytics capabilities in the form of service to customers in a pay-per-use economic model. In today's competitive world, the potential business values of these applications depend a lot on the quality of service offered by providers. Hence, to gain competitive advantages, providers need to be more customer-focused and proactive in their marketing strategies not only to create customers awareness of their services but also meet customers' best expectations for service quality. That is to say, providers must provide the required and promised services, and the services must satisfy users' requirements, such as availability, scalability, elasticity, and so on.

Given these circumstances, it is very important and necessary for efficient methods to manage and guarantee the quality of service promised. Service Level Agreement (SLA) that represents the contract between providers and customers which captures the agreed upon guarantees regarding the quality of service is one of the effective methods. SLAs play an integral role in governing the relationships between providers and customers in the context of these applications in cloud. Besides setting the expectations by dictating the quality and the type of service, SLAs are also increasingly considered as a strong differentiator allowing a provider to offer different levels of service guarantees and to differentiate itself from its competitors. Therefore, how to manage SLAs for cloud-hosted big data analytics application (BDAAs) in ensuring SLA guarantee has become a crucial and essential aspect.

However, the management of SLAs for cloud-hosted BDAAs is extremely challenging due to the increased complexities and uncertainties imposed by the applications. For instance, the applications usually span heterogeneous and distributed software frameworks across multiple layers, which considerably impacts the allocation and configuration of

datacenter resources in order to accommodate changes in the big data workloads and to guarantee analytic results within SLA constraints.

Most of the extant studies focus on SLAs management in traditional distributed computing environments like grid computing or cloud computing, which is not the case for SLA management for BDAAs in cloud. Although the research on the management of SLAs for BDAAs in cloud is now attracting growing attention, to the best of our knowledge, the study on SLA management for BDAAs in cloud is still in its infancy.

In this thesis, we focus on the research problem that how to manage SLAs for big data analytics applications in cloud in ensuring SLA guarantee. This research problem has been break down into five research questions that have been respectively addressed in Chapter 3 to Chapter 7. Accordingly, we have made five major contributions including one systematic literature review contribution, one conceptual contribution and three technical contributions.

To the best of our knowledge, this thesis is one of the first attempts to systematically study SLA management for big data analytics applications in cloud. It is believed that the outcomes of this research will yield very positive contribution in terms of technical content, significance, and impact to the advancement of scientific research in this field.

Contents

Acknowledgments **ii**

Abstract **iv**

List of Figures **xii**

List of Tables **xv**

1 Introduction **1**

1.1 Problem Identification 2

1.1.1 Weakness Analysis of SLA Management from Industry Aspect . 4

1.1.2 Weakness Analysis of SLA Management from Academia Aspect 7

1.2 Research Problem 8

1.3 Research Questions 8

1.4 Research Framework 10

1.5 Significance and Relevance 10

1.6 Structure of the Thesis 11

2 Background **13**

2.1 Big Data 13

2.1.1 Definition of Big Data 13

2.1.2 Characteristics of Big Data 14

2.2 Cloud Computing 15

2.2.1 Cloud Computing 16

2.2.2 Three Types of Cloud Services 16

2.3 The Relationship between Big Data and Cloud Computing 17
2.4 Cloud-hosted Big Data Analytics Applications 18
2.5 The Framework of Big Data Analytics as a Service 20
2.6 Service Level Agreements 25
2.6.1 Definition of Service Level Agreements and Its Relevant Terms . 25
2.6.2 The Evolution of SLA 27
2.7 SLAs for Cloud-hosted BDAAs 31
2.7.1 Why SLAs are important for Cloud-hosted BDAAs 31
2.7.2 Why are SLAs for BDAAs in Cloud sophisticated? 35
2.7.3 How is SLAs for BDAAs in Cloud different from the previous evolutionary stages? 35
2.8 Summary 36

3 Systematic Literature Review on SLA Management for Cloud-hosted BDAAS 38

3.1 Positioning versus Existing Surveys 38
3.2 The Application of SLR Methodology on SLA Management for Cloud-hosted Big Data Analytics Applications 40
3.2.1 Systematic literature Review Questions 40
3.2.2 Search Strategy for Primary Studies 42
3.2.3 Search Criteria 43
3.2.4 Selection Mechanism 44
3.2.5 Data Extraction and Synthesis 44
3.3 Review Results in Thematic Taxonomy of SLA Management for cloud-hosted BDAAs 47
3.3.1 Actors (In response to *SLRQ1*) 47
3.3.2 Service Layers (In response to *SLRQ2*) 48
3.3.3 Cloud Deployment Models (In response to *SLRQ2*) 55
3.3.4 Techniques (In response to *SLRQ3*) 58
3.3.5 SLA Metrics (In response to *SLRQ4*) 71
3.3.6 Conceptualization (In response to *SLRQ5*) 73

3.4 Summary 74

4 A New SLA Model and Categorization Scheme of SLA Metrics for Big Data Analytics Applications in Cloud 75

4.1 Design Principles and Requirements 75
4.2 Proposed Cross-layer SLA Model for Cloud-hosted BDAAs 77
4.3 Multi-dimensional Categorization Scheme of SLA Metrics for Cloud-hosted BDAAs 80
4.3.1 aSLA Metrics 80
4.3.2 pSLA Metrics 82
4.3.3 cSLA Metrics 82
4.4 Detailed Example of SLAs Template for a Real Cloud-hosted BDAA . . 83
4.5 Summary 90

5 Modeling and Simulation of Big Data Analytics Applications in Cloud 91

5.1 Positioning versus Existing Simulation Techniques 91
5.2 Modeling Requirements for Cloud-hosted BDAAs 95
5.2.1 The MapReduce Model 95
5.2.2 BDSaaS-level Requirements 97
5.2.3 BDPaaS-level Requirements 98
5.2.4 ClaaS-level Requirements 98
5.3 The Design Decisions and Architecture of IoTSim 98
5.3.1 Design Principles 99
5.3.2 Specific Design Considerations 100
5.3.3 The Architecture of IoTSim 101
5.4 The Development of Fundamental Classes and Their Correlations . . . 103
5.5 An Example of Simulation Using IoTSim 107
5.6 Evaluation 110
5.6.1 Experiment Setup 110
5.6.2 Independent Variables 111
5.6.3 Dependent Variables 114

5.6.4 Evaluation Scenarios . . . 116
5.7 Experiment Results and Analysis . . . 117
5.7.1 Variation of the Number of MapReduce Tasks . . . 117
5.7.2 Variation of the Number of VMs . . . 119
5.7.3 Variation of the Types of VMs . . . 120
5.7.4 Variation of the Job Types . . . 121
5.8 Summary . . . 122

6 SLA-driven Scheduling Big Data Analytics Applications in Public Cloud 124

6.1 Positioning versus Existing Scheduling Techniques . . . 124
6.2 Cross-layer System Model for SLA-driven Scheduling BDAAs in Public Cloud . . . 126
6.3 Mathematical Model and Assumptions on the Monetary Cost . . . 129
6.3.1 MapReduce Job Model . . . 129
6.3.2 Makespan of MapReduce Job . . . 131
6.3.3 Multi-objective Optimization Problems . . . 135
6.4 Proposed MapReduce Application Scheduling Algorithm . . . 135
6.5 Evaluation . . . 137
6.5.1 Benchmarking Algorithms . . . 138
6.5.2 Experimental Setup . . . 139
6.5.3 Evaluation Scenarios . . . 142
6.6 Analysis of Experimental Results . . . 142
6.6.1 Variation in the Number of Concurrent User Requests . . . 142
6.6.2 Variation in VM MIPS Rating . . . 144
6.6.3 Variation in Deadline . . . 144
6.6.4 Variation in Budget . . . 147
6.6.5 Variation in Jobs and Deadline . . . 147
6.7 Summary . . . 150

7 Detection of SLA Violation for Big Data Analytics Applications in Cloud **151**

7.1 Positioning versus Existing SLA Violation Detection Techniques 151
7.2 Problem Formulation of SLA Violation Detection 154
7.3 Dataset 154
7.4 Feature Extraction 157
7.5 SLA Violation Definition 160
7.6 Examples of Features and SLA Violation Status 161
7.7 Prediction Models 163
7.7.1 Logistics Regression 163
7.7.2 Artificial Neural Network 164
7.7.3 Random Forest 165
7.7.4 Extreme Gradient Boosting 167
7.8 Tackling Unbalanced Data 168
7.8.1 Oversampling Techniques 169
7.8.2 Undersampling Techniques 170
7.8.3 Combination of Oversampling and Undersampling Techniques . 171
7.9 Evaluation Metrics 171
7.9.1 Accuracy 172
7.9.2 Precision 173
7.9.3 Recall 173
7.9.4 F_β Score 173
7.9.5 Receiving Operating Characteristics (ROC) 173
7.10 Evaluation and Analysis of the Prediction Models 174
7.10.1 Results of Cross Validation 175
7.10.2 Analysis of Prediction Outcome 179
7.11 Mathematical Model and Assumptions on Providers' Profit 184
7.11.1 Admission Control Policy and Profit Loss Matrix 185
7.11.2 The Formulation of a Provider's Profit 188
7.12 Experiments and Discussion 189
7.13 Summary 194

8 Conclusion and Future Works **197**

8.1 Summary of the Research 197

8.2 Major Research Contributions 199

8.2.1 A Novel Thematic Taxonomy of SLA Management for BDAAs in Cloud Using Systematic Literature Review Methodology 199

8.2.2 A New SLA Model and General Categorization Scheme of SLA Metrics for BDAAs in Cloud 200

8.2.3 A New Simulation Toolkit for Modeling and Simulating BDAAs in Cloud 200

8.2.4 A New Cost-efficient SLA-driven Scheduling Algorithm for MapReduce applications 200

8.2.5 A Novel SLA Violation Detection for BDAAs in Cloud 201

8.3 Limitations and Future Works 201

Bibliography **204**

Publications during Ph.D. Study **241**

List of Figures

1.1 Research framework 9

2.1 The 3Vs. of big data 15
2.2 The framework of big data analytics as a service 21
2.3 SLA evolution stages with representative references 29
2.4 A stock recommendation BDAA hosted in AWS Datacenter 34

3.1 Systematic literature methodology applied on the survey of SLA management for cloud-hosted BDAA 41
3.2 Taxonomy of SLA management for cloud-hosted BDAAs 46
3.3 Statistics of the reviewed papers by layer and their breakdown 49
3.4 Statistics of the reviewed papers by cloud deployment models 56
3.5 Breakdown of public cloud deployment mode 57
3.6 Breakdown of private cloud deployment mode 57
3.7 Breakdown of hybrid cloud deployment mode 58
3.8 Statistics of the reviewed papers by techniques 60
3.9 Statistics of simulation technique used in the reviewed papers 68
3.10 Statistics on the frequency of SLA metrics in the reviewed papers 73

4.1 Categorization of requirements for layer-based BDAAs in cloud 76
4.2 Cross-layer SLA model for cloud-hosted BDAAs 78
4.3 Categorization scheme of SLA metrics for BDAAs in clouds 81
4.4 Architecture of Google's inventory system for retail 86
4.5 Cross-layer SLA metrics for Google's inventory system 90

5.1 The workflow of MapReduce operation 96
5.2 The MapReduce architecture in Hadoop 97
5.3 The proposed layered architecture of IoTSim 104
5.4 Class diagram of fundamental entities 105
5.5 The basic steps in the simulation of IoTSim 109
5.6 How the number of MapReduce tasks impact execution time and makespan: (a) execution time (Average, Max, Min); (b) Makespan 118
5.7 How the number of VMs impacts the average execution time 119
5.8 How the number of VMs impacts the total network cost 120
5.9 How the VM type impacts the average execution time 121
5.10 How the job type impacts the total VM cost 122

6.1 A high-level system model for scheduling BDAAs in public cloud 127
6.2 The compositions of makespan when executing a MapReduce job J 132
6.3 Variation in the number of concurrent user requests (a) Average Makespan; (b) Average Cost 143
6.4 Variation in VM MIPS rating (a) Average Makespan; (b) Average Cost . . 145
6.5 Variation in deadline (a) Average Makespan; (b) Average Cost 146
6.6 Variation in budget (a) Average Makespan; (b) Average Cost 148
6.7 Variation in jobs and deadline (a) Average Makespan; (b) Average Cost . . 149

7.1 Entity relationship diagram of batch workload information in the Alibaba datase 156
7.2 The structure of batch workload in Alibaba trace dataset 158
7.3 Heavily skewed dataset (violated and non-violated) 161
7.4 Computation and transformation process of a single neuron 164
7.5 Design of a 3-layer neural network for Alibaba trace dataset 166
7.6 Working mechanism of random forest 167
7.7 Working mechanism of XGB 168
7.8 Top 10 techniques measured by accuracy 180
7.9 Top 10 techniques measured by precision 181

7.10 Top 10 techniques measured by recall 182
7.11 Top 10 techniques measured by F_2 183
7.12 Top 10 techniques measured by ROC 184
7.13 Radar chart of the four prediction techniques based on the five evaluation metrics 185
7.14 The capability of a prediction technique in achieving profit on different $PenaltyRate$ and $ProfitMargin$ combinations 190
7.15 An example of selected single decision tree for RF and XGB respectively . 192
7.16 The average importance of each feature in RF and XGB respectively . . . 193
7.17 The aggregated feature importance by layer in RF and XGB 195

List of Tables

1.1 Comparative analysis of leading providers' SLA and their financial credit . 5

2.1 Summary on the BDAaaS framework by layer 24

3.1 Academic database sources 42
3.2 List of Inclusion Criteria 43
3.3 List of Exclusion Criteria 44
3.4 Classification of the reviewed papers on SLA Management for BDAAs in clouds by layers 51
3.5 Techniques used in addressing SLA management for BDAAs in clouds . . . 59
3.6 SLA metrics and their measurement for BDAAs in clouds 72

4.1 Common aSLA metrics 81
4.2 Niche aSLA Metrics by different types of BDAAs 82
4.3 Common pSLA metrics 83
4.4 Niche pSLA metrics by the different framework at BDPaaS layer 84
4.5 Common cSLA metrics 85
4.6 Niche cSLA metrics by different components at CIaaS layer 85
4.7 Example of aSLA, pSLA, and cSLA for Google's inventory system 89

5.1 Datacenter configuration 112
5.2 Host parameters 112
5.4 Job parameters 113
5.3 VM parameters 113
5.5 Notation used in description of dependent variables 115

6.1 Notation used in the mathematical model of the monetary cost 130

7.1 Notation used in formulating SLA violation detection problem 155
7.2 Summary of the ten features and their classification by layer 162
7.3 Samples of two classes (violated and non-violated Job) 162
7.4 Confusion matrix in binary classification 172
7.5 Cross validation result for logistics regression with default setting 176
7.6 Cross validation result for a 3-layer artificial neural network 177
7.7 Cross validation result for random forest with default setting 178
7.8 Cross validation result for XGB with default setting 179
7.9 Notation used in the mathematical model of providers' profit 186
7.10 Admission control policy matrix by providers 186
7.11 Profit loss matrix for SLA violation detection problem (Unit: dollars/job) . 187
7.12 Different combination of $PenaltyRate$ and $ProfitMargin$ 189

Chapter 1

Introduction

We are now entering an era of "The Fourth Industrial Revolution" also known as Industry 4.0, referred to by the world economic forum [1, 2, 3]. In this revolution, we will see breakthroughs thanks to technology advancements in areas including artificial intelligence, robotics, quantum computing, 3D printing and the Internet of Things (IoT). Another technological growth area underpinning the Industry 4.0 in recent years has been big data [4, 5, 6, 7, 8], and it is changing every sector and industry regardless of size. The term big data (BD) refers to technologies and techniques that involve data that is massive, heterogeneous and fast-changing for conventional technologies, skills, and infrastructure to address efficiently. With the power and potential of exploring valuable information and ascertaining valued insights for the enhanced decision-making process, BD has become an essential technology driver of Industry 4.0 [6, 9]. In manufacturing, for example, enhancement and efficiencies in the decision making upon the information, evidence, and insight acquired by BD are expected to bring billions of dollars to the industry over the next five years [5, 10].

In the past decade or so, BD has seen exponential growth. On the one hand, BD is generated from different sources, i.e., web transactions, mobile devices, social media, and camera in the form of logs, text, voice, images, and video. According to a study by IBM, 2.5 quintillions bytes of data every day as of 2012 have been created [11]. IDC, the global market firm, estimated that in 2011 human created 1.8 zettabytes information [12]. They go on to forecast that 2.8 zettabytes were created in 2012. Globally, approximately 1.2 zettabytes of electronic data are generated per year by various sources [13]. Further, by 2020, enterprise data is expected to total 40 zettabytes,

as per IDC [8]. On the other hand, BD is generated by an expanding IoT [14, 15, 16] via which large amounts of data is collected, i.e., sensors, machines, vehicles, and wearable devices. The installed base for the IoT will grow to approximately 212 billion devices by 2020, which includes 30.1 billion installed connected autonomous things [17, 18, 19].

Organizations can leverage big data analytics technologies on such vast data to extract valuable information, gain actionable insight and enjoy the tangible value of data. Even though data collected by individual devices may not provide sufficient information, aggregated data from a number of physical devices and virtual sensors (e.g. social media such as Facebook, Twitter) [20, 21] can provide a wealth of knowledge for essential application areas including disaster management, sentiment analysis, smart cities, and so on.

However, BD has posed substantial technological challenges including data storage, data transmission, data processing, data analysis, data visualization and so on [9, 22, 23, 24]. These challenges are non-trivial and require the flexibility, scalability, and capability of storage and computing resources. Cloud Computing (CC) [25, 26, 27] could be leveraged to tackle these challenges in Industry 4.0 as it can offer on-demand, large and scalable computing and storage resources [22, 28]. This is because, on the one hand, BD is relatively large and complex which imposes unprecedented demands on the underlying infrastructures. On the other hand, public clouds, emerged in recent years, offer massive resources available and allow general users to rent virtual machines (VMs) to run their BDAAs in a pay-as-you-go manner with ultra-high scalability and yet minimized upfront costs.

1.1 Problem Identification

Cloud computing seems to be a perfect vehicle for hosting big data analytics applications (BDAAs). However, working on these BDAAs in cloud brings its own issues [23, 29]. Primarily, in this scenario, providers like Amazon, Google, and Microsoft provision big data analytics capabilities in the form of service [29, 30, 31, 32] in cloud to customers who consume this service in the pay-as-you-go pricing model. Thus, it is reasonable that customers who pay for such service expect it to be (among other

features) available, reliable, secure and so on. Customers could acquire the real value of BDAAs and benefit from knowing to what extent providers deliver the promised service. However, for cloud-hosted BDAAs, customers have less control of the actual service and the level of service offered by providers in compassion with developing and deploying the applications in an on-premise environment. In today's competitive business environment, the potential business values of BDAAs depend a lot on the quality of service (QoS) and providers must provide the required and promised services, and the services must satisfy customers' requirements. The unsatisfactory service level of big data analytics service may lead to increased service time and so to higher costs charged to customers. As a consequence, providers will fail to achieve competitive advantages.

Therefore, an essential and tough issue arises that is the management of service level agreement (SLA) for cloud-hosted BDAAs to ensure SLA guarantee. Such that, BDAAs are behaving as expected and agreed, for instance, exhibiting the required and agreed performance, guaranteeing a minimum precision, preserving privacy and security properties, and satisfying users' requirements such as on-demand availability, scalability, elasticity, security and so on. Actually, SLAs are increasingly considered as a critical differentiator allowing a provider to offer different levels of service guarantees and to differentiate itself from its competitors [33, 34, 35]. SLA is an efficient method to guarantee the level of service and plays an integral role in governing relationships between providers and customers.

However, the management of SLA for cloud-hosted BDAAs is extremely challenging due to the increased complexities and uncertainties imposed by the applications. For instance, BDAAs usually span heterogeneous and distributed software frameworks across multiple layers, which considerably impacts the allocation and configuration of datacenter resources in order to accommodate changes in the workload and to guarantee analytic results within SLA constraints. Also, the provider and customer's staff are more involved in the loop than in the traditional cloud environment [36]. Moreover, more components across multiple layers with contrasting and competing resources requirements interweave and interact for provisioning the applications to customers, which brings complicated SLA dependency relationship and hidden patterns

to be uncovered. Also, the use of real infrastructures such as Amazon EC2 limits the experiments to the scale of the infrastructure and makes the reproduction of results an arduous undertaking.

Although this research area is now attracting growing attention, the study on SLA management for cloud-hosted BDAAs is still in its infancy both in industry and academia, which will be elaborated in the following sections.

1.1.1 Weakness Analysis of SLA Management from Industry Aspect

In recent years, many providers offer a variety of big data analytics services in cloud to users on a pay-per-use basis, such as Google BigQuery [37], Amazon Redshift [38] and Microsoft Data Lake Analytics [39]. However, these leading providers offer services in the "best-effort" way, few of them deliver a solid SLA guarantee. Usually, they only commit to guarantee availability under the presence of hardware failures. For instance, even a company as large as Google offers only 99.9% guaranteed Monthly Uptime Percentage (availability) for their Google BigQuery service [40], yet some organizations require 99.99% uptime. If service uptime drops to 99.0%-99.9%, per Google's agreement customers are eligible for a service credit equal to 10% of their bill [40]. On November 2016, Google BigQuery faced a four-hour outage resulting in 73 percent of BigQuery streaming inserts failure with a 503 error code (an HTTP response status code indicating that a server is temporarily unable to handle the request). At the peak of the problem, the failure rate was 93 percent [41, 42]. Google recognized that the impact of this outage was substantial and it caused significant disruption for those customers who rely on the products which were part of this event. Google provided those impacted customers an SLA credit for the affected timeframe according to the SLA contract [41, 42]. Google BigQuery services do not provide performance guarantees or other SLA guarantees. Its competitive services provided by Amazon, and Azure exhibit similar behaviors.

Further, we examine SLAs documentation offered by some market-leading providers and compare their SLA metrics shown in Table 1.1.

Table 1.1: Comparative analysis of leading providers' SLA and their financial credit

Service Category	**Service Name**	**SLA Metrics**	**Financial Credit**
Data Processing	Google Cloud Dataflow [43] or Google Cloud Dataproc [44]	Monthly Uptime Percentage	• 99% - < 99.5% : 10% • 95% - < 99% : 25% • < 95% : 50%
	Microsoft Azure HDInsight [45]	Monthly Uptime Percentage	• 99% - < 99.9% : 10% • < 99% : 25%
	Amazon Elastic MapReduce [46]	N/A	N/A
	Rackspace Hadoop Service [47]	Monthly Availability	• 99.5% - < 99.9% : 10% • 99% - < 99.5% : 25% • < 99% : 30%
	SAP Cloud Platform Big Data Service [48]	System Availability percentage	• < 99.5% : 2% of monthly subscription fees for each 1% below system availability
	Alibaba Elastic Compute Service [49]	Monthly Uptime Percentage	• 99% - < 99.95% : 10% of the Monthly Service Fee per Instance • 95% - < 99% : 25% of the Monthly Service Fee per Instance • < 95% : 100% of the Monthly Service Fee per Instance
Data Analytics	Google BigQuery [40]	Monthly Uptime Percentage	• 99% - < 99.9% : 10% • 95% - < 99% : 25% • < 95% : 50%
	Microsoft Data Lake Analytics [50]	Monthly Uptime Percentage	• 99% - < 99.9% : 10% • < 99% : 25%
	Amazon Redshift [38]	N/A	N/A

Based on Table 1.1, we could deduce that the weaknesses of SLA management for cloud-hosted BDAAs fall into the following categories.

- **Lack of standard format or presentation**: Providers typically published SLAs on their websites in multifarious formats which varies significantly with each other. Moreover, the relevant information may be updated without prior notice to end users. Hence, it is not easy and often time-consuming for customers to obtain and compare SLAs information from their websites and documentation.

- **Non-standardized terms used among providers**: Take one of very common SLA metrics availability as an example, this term used by providers is not consistent. Specifically, Amazon, Microsoft, and Alibaba refers availability as Monthly

Uptime Percentage, while SAP refers to the System Availability Percentage, and Rackspace uses Monthly Availability.

- **Incompleteness metrics**: SLAs in these providers' products usually cover the availability and performance of the underlying infrastructure, leaving the SLAs at platform-level software (e.g., Hadoop cluster) and application-level (e.g., smart city, social network driven stock recommendation) largely untouched. Take Microsoft Azure HDInsight service as an instance, in this public SLA documentation, Microsoft only mentions availability (uptime) there, while ignoring or missing measurable assurances or guarantees on other SLA metrics like service response time, makespan, throughput or even energy consumption. However, there are vital factors that are not included or very imprecisely described in the currently existing SLAs. We cannot deny that availability is one of the critical qualities that need to be included in an SLA, but it is not the only one. Most importantly, big data analytics applications rely on many components (for example, data storage, data processing, etc.,) working together to meet SLAs. Each component has a different nature and features different SLA metrics.

- **Ambiguous language**: SLAs templates are written in a natural language which is ambiguous to consumers. Some providers left the technical terms unexplained which cause misperception to decide on SLA. For example, Google BigQuery service level agreements [40] specify that "Downtime means more than a five percent error rate. Downtime is measured based on server side error rate". However, the calculation on the server side error rate depends on the back-off requirements which insist back-off interval of one second for each successive requests that further reduce the downtime [51]. Here, Google defines that error rate is the number of valid requests that result in a response with HTTP Status 500 and code "Internal Error" divided by the total number of valid requests during that period. Back-off requirements mean when an error occurs, the application is responsible for waiting for a period of time before issuing another request [40].

1.1.2 Weakness Analysis of SLA Management from Academia Aspect

Over the years, a large body of research on SLA management exist. Most of these have been conducted in traditional network environment, cluster computing environment, grid computing environment, or cloud computing environment. To the best of our knowledge, very few research specifically address the management of SLA for cloud-hosted BDAAs. The weakness from academia consists of the following aspects:

- **Lack of SLA model for cloud-hosted BDAAs**: There are different perspectives examining SLAs for cloud-hosted BDAAs. Some discuss SLAs in the cloud infrastructure level, some focus SLAs at the application level, while others observe SLAs at the platform level. These different perspectives result in various SLAs with distinct features and attributes, which cause ambiguity to customers. Moreover, the studies on SLAs are relatively isolated and scattered among many papers. There is a lack of an SLA model to describe and model SLAs in a unified and structured way and demonstrate complex SLA dependencies relationships across layers while respecting the characteristics of big data analytics application.

- **Absence of a general categorization scheme of SLA metrics in cloud-hosted BDAAs**: The study on the categorization of SLA metrics has been considered by many researchers in telecommunication, grid computing, and cloud computing environment. However, SLA metrics in these particular environments are either oversimplified or inappropriate for cloud-hosted BDAAs as the nature and type of resources being provided and delivered is entirely different. Essentially, cloud-hosted BDAAs have a complex architecture where various components at different level interact and possess distinct and competing SLA metrics. The study on the categorization of SLA metrics for cloud-hosted BDAAs is very scarce so far.

- **Shortage of systematic literature review on SLA management for cloud-hosted BDAAs**: A few studies have been performed on the review of SLA management in cloud environment. However, to the best of our knowledge,

a recent systematic literature review specifically on the SLA management for cloud-hosted BDAAs is lacking. As a result, a thematic taxonomy that gives a comprehensive outlook in this research field is missing.

- **Shortcomings of extant SLA management approaches**: Many approaches have been leveraged to successfully address SLA management in cloud environment. However, these approaches cannot be straightly used for cloud-hosted BDAAs due to characteristics and complexities of BDAAs. The SLA management in this context should be systematically investigated, identified and improved to tackle the issue relevant to cloud-hosted BDAAs.

1.2 Research Problem

The above analysis of SLA management weakness from both industry and academia aspects leads to our identified research problem: *How to manage service level agreements for big data analytics applications in cloud in ensuring SLAs guarantee?*

1.3 Research Questions

To tackle our research problem in a more step-by-step manner, we break down the above research problem into the following sub research questions (RQs):

i What is the landscape of the extant research on the management of service level agreements for cloud-hosted big data analytics applications and how to classify them from different perspectives?

ii How to describe and model service level agreements for cloud-hosted big data analytics applications (BDAAs) in a unified and structured way, and specify common and niche SLA metrics respecting characteristics of BDAAs?

iii How to model and simulate cloud-hosted big data analytics application (BDAAs) across layers to facilitate SLAs management?

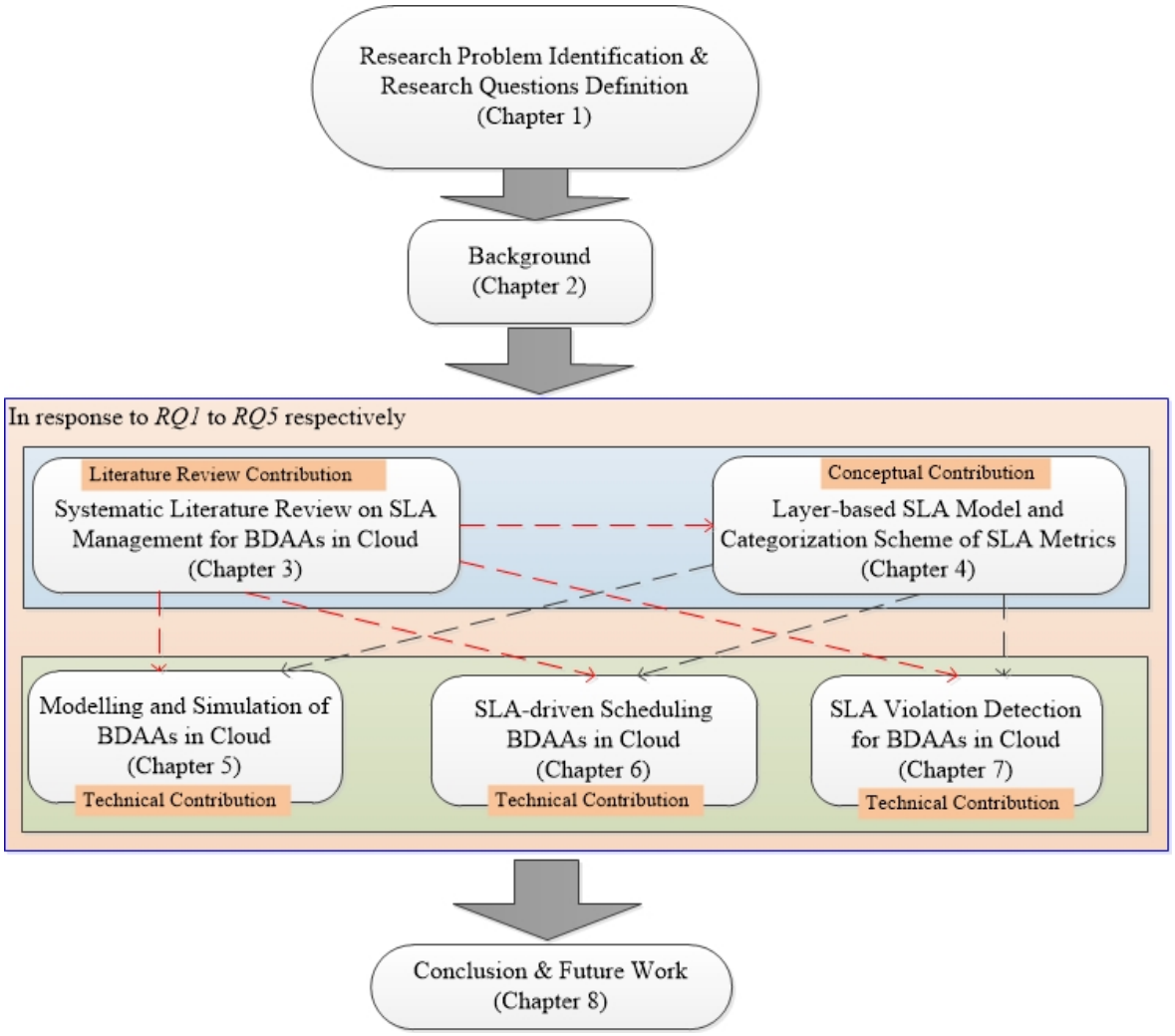


Figure 1.1: Research framework

iv How to achieve SLA-driven scheduling for cloud-hosted big data analytics application (BDAAs) across layers while guaranteeing SLA?

v How to detect SLA violations for cloud-hosted big data analytics application (BDAAs) across layers before they happen to maximize providers' profit?

1.4 Research Framework

This section discusses the logical research framework shown in Figure 1.1 to ensure that our research processes and the proposed solutions are addressed using a systematic and scientifically based method. Because this research involves a combination of diverse research methods including systematic literature review, conceptual model design, software design and implementation, algorithm development and so on to our identified research problem, we discuss and justify research methods chosen as the most suitable for our study. According to Denzin and Lincoln, research methods are skills and practices that provide guidelines for researchers to address research questions and the research process [52]. They discuss the scope of problems, the research resources and the rules for researchers when they undertake research projects. Systematics literature methods, case studies, software engineering methods, model design methods, experimental methods, simulation methods, and mathematical modeling methods are examples of research methods for computer science [53]. The use of more than one type of research method (all known as mixed method) is feasible and often adopted in specific combinations [54, 55]. Since this thesis aims to explore the mechanism regarding how to manage the service level agreements for big data analytics applications in cloud in ensuring SLA guarantee. We intend to combine multiple research methods to tackle our identified research problem, such as (i) defining fundamental concepts and terms; (ii) designing conceptual model; (iii) using systematic literature review methodology to explore thematic taxonomy; (iv) using soft engineering method to design and implement simulation tool; (v) designing mathematical model and developing algorithms.

1.5 Significance and Relevance

We have seen that there is a well-established increasing trend that the adoption of big data analytics applications (BDAAs) in cloud and hence importance and urgent necessity on the management of service level managements (SLAs) for cloud-hosted BDAAs. To the best of our knowledge, this Ph.D. research is one of the first attempts to systematically study how to manage SLAs for BDAAs in cloud. We have made five

contributions including one systematic literature review contribution, one conceptual contribution and three technical contributions. The taxonomy, conceptual model and techniques proposed in the thesis are not only significant in theory but also in practice. It is believed that the outcomes of this research would generate very positive contribution in terms of technical content, significance, and impact to the advancement of scientific research in this field of computer science.

1.6 Structure of the Thesis

The rest of this thesis is structured as follows.

Chapter 2 presents the background of our work. It contains some fundamental terms and concepts of big data, cloud computing, big data analytics applications and critical information regarding service level agreements. Moreover, in this chapter, why SLA for cloud-hosted BDAAs is imperative and remains a sophisticated challenge, and how SLA for cloud-hosted BDAAs is different from SLA for cloud computing has been analyzed.

Chapter 3 contains the first contribution of this thesis by using a systematic literature review to conduct an in-depth survey on the SLA management for cloud-hosted BDAAs. We propose a new and comprehensive thematic taxonomy, which brings benefits for future researchers to quickly understand the whole picture and identify research gaps in this research field.

Chapter 4 describes the second contribution with two aspects. Firstly, a novel cross-layer SLA model for cloud-hosted BDAAs has been proposed, which provides a unified and structured way to understand SLAs at different layers with different attributes and their strong dependencies relationship. Secondly, a general categorization scheme of SLA metrics is proposed, consisting of common metrics and niche metrics for each different level SLA, which fully embodies characteristics of cloud-hosted BDAAs.

Chapter 5 discusses the modeling and simulation of cloud-hosted BDAAs, which leads to the third contribution. In particular, the modeling requirements for cloud-hosted BDAAs has been analyzed, and a novel simulator has been designed and implemented. This simulator can mimic the behaviors of cloud-based MapReduce applications, analyze the impact of SLA constraints and enable researchers to evaluate their approaches or

solutions if meet SLAs requirements for BDAAs in cloud.

Chapter 6 addresses how to optimally schedule cloud-hosted BDAAs across layers while ensuring SLA guarantee, which makes the fourth contribution. Specifically, a new SLA-driven algorithm of scheduling cloud-hosted BDAAs that considers users' SLA constraints has been designed and developed. This new scheduling algorithm can minimize the cost of renting cloud resources from public cloud providers to concurrently process multiple applications workloads using MapReduce model while satisfying SLA requirements.

Chapter 7 addresses how to use machine learning and resampling techniques to successfully detect SLA violations for cloud-hosted BDAAs in Alibaba's batch workloads dataset, which are the fifth contribution. Importantly, the hidden patterns of the multiple configurations across layers are uncovered, and insightful information is provided for providers' decision making.

Chapter 8 firstly summarizes our work and then highlights five major contributions. Last, we discuss the limitations of our work and points out possible future works.

Chapter 2

Background

2.1 Big Data

2.1.1 Definition of Big Data

While it is ubiquitous today, big data (BD) does not have an absolute, precise and agreed upon definition [56, 57, 58].

Amongst the most cited definitions is that included in a META Group (now Gartner) report written by Doug Laney as far back as 2001 [59]. He proposed a 3D data management concept encompassing 3Vs: Volume, Velocity, and Variety. The report remarks upon the increasing size of data, the increasing rate at which it is produced and the increasing range of formats and representations employed. Although the phrase "big data" is not explicitly mentioned in this report, the report has since been coopted as a key definition. To the best of our knowledge, this is the earliest definition of BD. From that point forward, many academia and practitioners give more definitions of BD.

In 2011, an IDC report defined BD as "a new generation of technologies and architectures, designed to economically extract value from huge volumes of a wide variety of data, by enabling high-velocity capture, discovery, and analysis." [12]

In 2012, Gartner reiterated the definition: "big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making." [60]

In 2015, U.S. National Institute of Standards and Technology (NIST) referred BD

the need to parallelize data handling in data-intensive [61]. The characteristics of BD that force new architectures are Volume, Velocity, Variety, and Variability.

Some leading industry vendors also contribute to the definition of BD. For example, IBM defines that BD is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage, and process the data with low-latency [11]. Moreover, similar to the previous definition, it has one or more of the following characteristics – high volume, high velocity, or high variety. Microsoft provides a notably succinct definition stating that "big data is the term increasingly used to describe the process of applying serious computing power – the latest in machine learning and artificial intelligence – to seriously massive and often highly complex sets of information" [58]. This definition states in no uncertain terms that BD requires the application of significant computing power [62].

2.1.2 Characteristics of Big Data

Although the above big data (BD) definitions are diverse, an agreement on the common characteristics of BD has been reached. The Volume, Variety, and Velocity (also known as 3Vs) have emerged as a universal model to describe the characteristics of BD [63, 64, 65]. Figure 2.1 shows the 3Vs model.

Volume refers to the magnitude of data that organizations are trying to harness to improve decision-making across the enterprise. Data volume is reported in multiple terabytes and petabytes. It is worth to mention that definitions of BD volume is relative and varies by factors, such as time, the type of data, industry and even geography. Moreover, it can be smaller than the petabytes and zettabytes often referenced in articles and statistics.

Variety has a twofold meaning. On the one hand, variety describes different formats or types of data that do not lend themselves to storage in traditional structured relational database systems. These include a long list of data such as spreadsheets, emails, social media text messages, images, video, audio and more. This type of data is characterized as unstructured and has existed all along. On the other hand, variety is also used to mean data from many different sources, both inside and outside of organizations. For

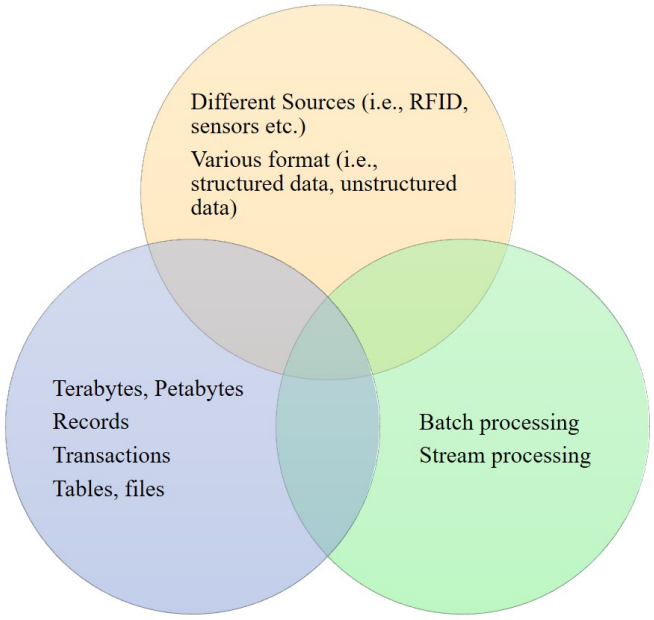


Figure 2.1: The 3Vs. of big data

example, data come from sensors, smart devices, RFID tags.

Velocity refers to the rate at which data are generated and speed at which it should be analyzed and acted upon. Nowadays, large data volumes are daily generated at an unprecedented rate from heterogeneous sources (e.g., health, government, social networks, marketing), which is driving a growing need for real-time analytics and knowledge discovery.

2.2 Cloud Computing

Cloud computing (CC) has attracted significant momentum and attention in both academia and industry over the past decade. However, it seems that no standard definition of CC has been widely accepted yet [66, 67, 68]. In this thesis, we adopted and considered the definition of CC issued by NIST on September 2011 as "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage,

applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [69].

2.2.1 Cloud Computing

One of the critical elements of cloud computing is the deployment model. According to NIST [69], four common cloud deployment models are stated as follows.

Public cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of cloud providers.

Private cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple customers (e.g., business units). It may be owned, managed, and operated by an organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of organizations in a community, a third party, or some combination of them, and it may exist on or off premises.

Hybrid cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

2.2.2 Three Types of Cloud Services

In general, cloud computing offers three most common types of services to end user [70, 71, 69, 67]. They are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

SaaS: A software licensing and delivery model where software is licensed on a subscription basis and is centrally hosted. SaaS providers offer an actual working software application to clients and have complete control of application software. They

hide the underlying details of software and provide an interface to work on the system. SaaS is a common delivery model for many applications.

PaaS: This type of cloud service offers a pre-built and managed application platform to customers. Customers need not spend time building underlying infrastructure for their applications. On the backend, PaaS automatically scales and provisions required infrastructure components depending on application requirements. Typically, PaaS solutions provide an API that includes a set of functions for programmatic platform management and solution development. This type of cloud service typically might contain an operating system, database management system, hosting, and data processing framework.

IaaS: This kind of cloud service gives low-level abstractions of physical devices. A cloud provider hosts infrastructure components traditionally present in an on-premises datacenter, including servers, storage, and networking hardware, as well as the virtualization or hypervisor layer. IaaS customers access resources and services through the internet and can use the cloud provider's services to install the remaining elements of an application stack.

2.3 The Relationship between Big Data and Cloud Computing

Taken separately, both big data (BD) and cloud computing (CC) can prove very beneficial to a company, providing numerous advantages in an increasingly competitive business environment. BD is an approach for maximizing the linear scalability, deployment and execution flexibility, and cost-effectiveness of analytic data platforms [72]. CC is the new era of computing where efficient utilization of resources can be done with no compromise on data size, execution time and cost of execution. CC complements BD by enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or interaction [69].

Essentially, BD and CC are two technologies that are often conjoined and increasingly incorporated together [65, 73]. Big data analytics applications (BDAAs) need process

BD which are relatively large and compound, and that requires the support of unique and advanced data storage, management, analysis, and visualization technologies [74, 75]. Hence, BDAAs requires lots of IT resource if users want a fast analysis of BD. Thousands of CPUs, hundreds of terabytes storages and very high-speed interconnections are demanded. For the fast analysis of BD, much more computing resources are required. Clouds are already deployed on pools of server, storage, and networking resources and can scale up or down as needed [76]. Therefore, it makes sense that organizations should look to CC as the resource provisioning platform to support their BDAAs.

While each technology is valuable on its own, many businesses are attempting to combine them to reap even more benefits. As CC has increasingly become a de facto part of doing business, more companies are choosing to implement and deploy and operate BD in cloud. By utilizing the cloud for this purpose, businesses can derive better analysis from the massive amounts of structured and unstructured data in their possession. This attribute of the cloud has proven so valuable that BD is even driving the adoption of CC across numerous industries and enterprises. CC is much cheaper for companies to use when compared to the large-scale BD resources that organizations have used before. CC also makes it much easier for companies to integrate data from numerous different sources and databases and can produce better results with a more consistent performance.

2.4 Cloud-hosted Big Data Analytics Applications

The potential of big data (BD) is unleashed when it becomes possible to mine analysis and insights from it. This is where big data analytics comes into the picture. The rise of big data analytics offers vast opportunities for existing organizations as well as for new start-ups, and both public and private organizations to construct valuable information from BD [77, 78].

According to [23, 29, 79], big data analytics is the method and process for examining a significant amount of datasets that contains a variety of data types and transforming them into a more understandable data and metadata format in order to reveal unseen patterns, hidden correlations, market trends, customer preferences, and other useful

business information. After analyzing data, correct tools and approaches visualize the findings in tables, graphs, and spatial charts for efficient decision making. The primary objective of big data analytics is to assist organizations to have an improved understanding of data, and thus, make efficient and well-informed decisions.

The power of big data analytics is usually released through big data analytics applications (BDAAs) that are a new type of software applications, which analyze BD using massive parallel distributed processing frameworks (e.g., Hadoop). Big data analytics has found applications across a broad range of domains such as banking and financial services, telecommunications, travel and entertainment, digital media, healthcare, manufacturing, and others [80, 81].

Leading providers offer a wide array of different types of BDAAs. For example, Google utilizes Google BigQuery [37] to offer inventory management system [82], an abundant, highly scalable, low cost and pay-as-you-go cloud-hosted BDAA to make inventory management productive and efficient. Amazon provides natural language processing-based BDAA in cloud that identifies the language of voluminous texts, extracts vital entities such as people, organizations, locations or events, and analyze sentiments in texts using Amazon Comprehend [83].

The provision of such BDAAs in cloud could ease adoption for many companies and in addition to substantial cost savings. It could simplify useful insights that provide them with different kinds of competitive advantage. In fact, more organizations seem to be opting for cloud nowadays. According to the blog post written by Brian Hopkins, Forrester vice president and principal analyst in August 2017 [84] "Global spending on big data solutions via cloud subscriptions will grow almost 7.5 times faster than the on-premise subscriptions. Furthermore, the public cloud was the number one technology priority for big data according to our 2016 and 2017 surveys of data analytics professionals". Also, a business intelligence to Hadoop/big data connection company AtScale [85] conducted a three-year survey on how global companies use BD and the cloud. The survey findings unsurprisingly revealed that cloud is effectively taking center stage for BD use. The survey has found that there has been an increased amount of attention to the deployment of BDAAs in cloud, with 59% of respondents has deployed BDAAs in cloud already. Additionally, over three quarters (77%) are

projecting they would use the cloud for BD up 5% from last year. These demonstrate that cloud-based BDAAs is a widespread phenomenon nowadays.

From the service's perspective, the capabilities and benefits of BDAAs are uniquely placed and delivered in the form of Big Data Analytics as a Service (BDAaaS) for marketing purposes. We will give a detailed description of BDAaaS in the next subsection.

2.5 The Framework of Big Data Analytics as a Service

In previous research works, cloud was considered as a marketplace [68, 86], where the storage and computing capabilities of cloud-based system architectures can be leased. Likewise, the rise of big data (BD) has created new business models, where businesses lease big data analytics service in a pay-as-you-go and service-oriented manner. In this direction, a new service named Big Data Analytics as a Service (BDAaaS) was introduced [86, 87].

BDAaaS represents a combination of analytics software and cloud technology to an extensible platform that can provide analytics capabilities over a variety of industries and use cases. It is a cloud-based spectrum of hardware and software services for storage and analysis of increased amounts of diverse data which have emerged in the past few years due to technological advancement and intrinsic presence of technology usage in everyday life [23, 29, 30]. Compared to hosting any data analytics software on-premises servers, BDAaaS offers solutions that are easy to deploy and most of the time has a pay-as-you-go payment system. It is part of a broader "as-a-Service" solutions such as SaaS/PaaS/IaaS. Unlike them, BDAaaS is an end-to-end service that leverages advanced technologies regarding data ingestion, data processing, data storage, and data analysis as well as management and maintenance tasks across layers.

Inspired by the work in [73, 88, 89, 90, 91], we give a pictorial representation of BDAaaS framework. The BDAaaS framework is designed with a flexible architecture describing as a set of connected components, presented and categorized by level of abstraction. When a BDAaaS instance is deployed in cloud, each abstract component

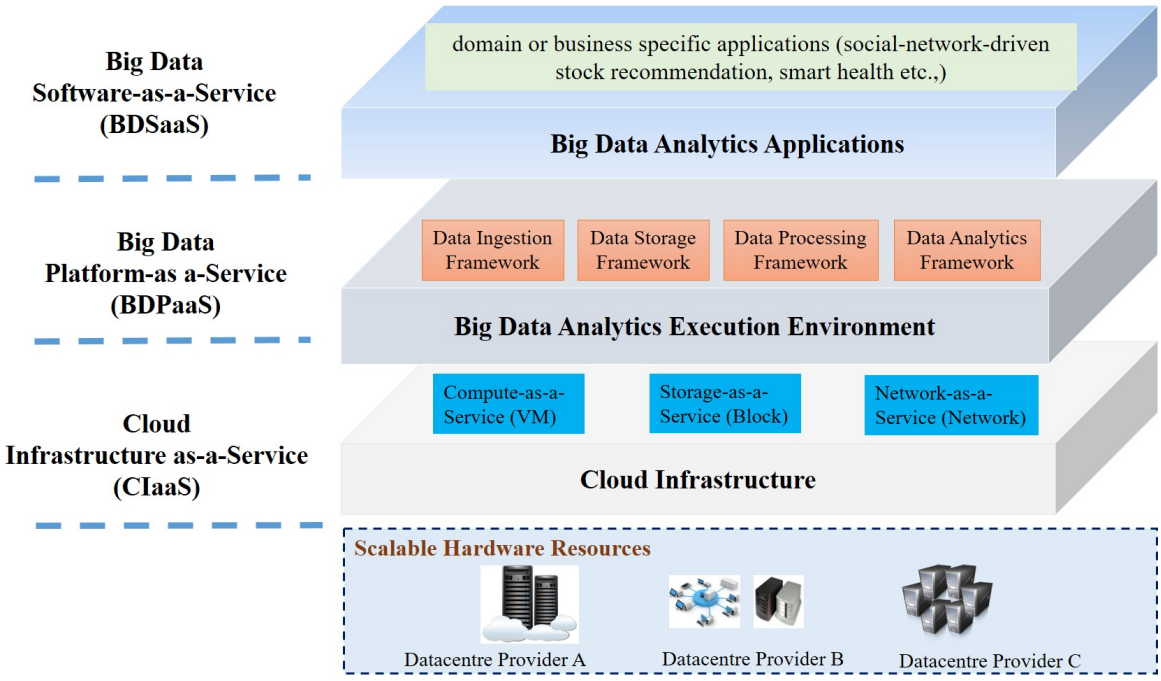


Figure 2.2: The framework of big data analytics as a service

will be specialized into a concrete component. The overall framework of BDAaaS is depicted in Figure 2.2.

We can observe in this figure three levels from top to bottom: Big Data Software as a Service (BDSaaS), Big Data Platform as a Service (BDPaaS) and Cloud Infrastructure as a Service (CIaaS). Beyond the top level are usually end users who request analytics service through the interface. It is not difficult to understand that an end user is a client of the BDSaaS, which is itself a client of the BDPaaS, which is itself a client of the CIaaS. Under the level of CIaaS exists scalable hardware resources in datacenters. The details of each level are described as follows.

Cloud Infrastructure as a Service (CIaaS): Cloud computing plays the role of underlying infrastructure in this framework located at the bottom layer. In this layer, users access three main elastic resources (i.e., computing resources, storage resources, and network resources) in a pay-as-you-go manner. This layer provides the most opportunities for direct influence on big data technology (scalability, availability,

computing, and accessibility of raw data). Amazon Elastic Compute Cloud (Amazon EC2) [92] is an excellent choice in this layer offering high-performance and on-demand computing resources, analogous to a server in a datacenter. Amazon Simple Storage Service (Amazon S3) is storage resources analogous to an array of hard drives [93].

Big Data Platform as a Service (BDPaaS): One level up exists the platform layer including software, tools or frameworks dedicated to BDAAs. For instance, Hadoop and its ecosystem [94]. BDPaaS emphasizes on the provision of a specialized execution environment for processing the applications, which are often instantiated over CIaaS to ensure scalability, availability, reliability, and so on. BDPaaS typically includes multiple frameworks/components as follows.

- Data Ingestion: This is the first step for the data coming from variable sources to start its journey, which deals with technologies for the ingestion of data into BDAAs. The main distinct dimension is velocity. This leads to two types of ingestion, i.e., batch and stream. Batch ingestion includes traditional Extract-Transform-Load data integration tools while stream ingestion is sub-divided into messaging systems and complex event processing engines [88]. Sample technologies for batch ingestion are Apache Sqoop [95]. Apache Kafka is a good example of stream ingestion [96].

- Data Storage: Represents technologies for persistent data storage for BDAAs. The main differentiating dimensions are volume and variety [88]. Variety distinguishes between different types of storages, i.e., distributed file systems (e.g., Hadoop HDFS [97], Google File System [98]), NoSQL (e.g., MongoDB [99], Hadoop HBase [100]) data stores, and traditional RDBMS (e.g., MySQL, Oracle etc.,). RDBMSs are mature and well support structured data, while unstructured data prefer NoSQL or distributed file systems, which is a reasonable choice for BDAAs.

- Data Processing: Includes technologies that are responsible for the execution and computation of voluminous data from data ingestion framework. The main differentiating dimensions are velocity and variety. It is mainly sub-divided into two categories consisting of batch processing and stream processing paradigm.

In the batch-processing paradigm, a large volume of data is first stored and then processed and analyzed at once [90]. The algorithm exploits an efficient divide-and-conquer approach that split the dataset into chunks and processes each of them individually on its own machine to generate intermediate results which are eventually aggregated to a final result. Apache Hadoop MapReduce [101] has become the dominant batch processing model. The streaming processing paradigm refers to the technology that queries continuous data streams and detects conditions quickly from the time of receiving the data at the second or even millisecond level. Apache Spark [102] and Apache Storm [103] are representative examples of stream processing.

- Data Analytics: Comprises technologies responsible for the value generation which is the ultimate objective of BDAAs. Such technologies uncover hidden patterns and unknown correlations and extract insight to improve decision making [90]. This framework is differentiated by two dimensions: the type of data analytics and the type of approaches. The type of data analytics comprises descriptive, predictive and prescriptive [36, 78, 104, 105]. The type of approaches includes statistics-based method [106, 107, 108] and machine learning-based methods [108, 109, 110]. Some representative examples of BDSaaS include Google BigQuery [37], Amazon Redshift [38] and Microsoft Data Lake Analytics [39].

Big Data Software as a Service (BDSaaS): This top-tiered layer typically provides specific BDAAs interfaces that enable users to focus on one particular domain of business or private concern and do not mention any underlying Cloud resources nor BDPaaS level components. Typically, BDSaaS is a web-based and multi-tenant system that analyzes and interprets massive amounts of data to deliver more insightful results for their subscribers. Users would develop and execute scripts and queries to analyze and generate reports and visualizations [89]. BDSaaS is an extension of the common SaaS model, with the same delivery model benefits, but better because BDSaaS leverages the best of the underlying BDPaaS-level technologies (i.e., data ingestion, data storage, data processing, and data analysis) in cloud to construct valuable information and gain real insight for users. Examples of BDSaaS include social media monitoring, analytics and

listening services like Salesforce.com's Marketing Cloud [111] or BrandsEye [112]. These BDSaaS can collect social media data in close to real-time and run in it through their underpinning BD technologies and algorithms to produce insights such as near-instant feedback on the effectiveness of new marketing campaigns or alerts about emerging problems with products [113].

Table 2.1 gives a summary of the BDAaaS framework. Compared to traditional on-premises hosting environments such as dedicated server farms, the framework is more modular. Each layer is loosely coupled with the neighboring layers, allowing the flexibility to evolve separately. The framework integrates cloud computing (CC) that acts the underlying infrastructure in the whole architecture, which highlights the integral role that CC plays in BDAAs. It is worthwhile to mention that although cloud is a natural architecture and popular choice for BDAaaS, this service is not limited to just cloud architecture. Other distributed architecture can also be employed to host big data analytics services.

Table 2.1: Summary on the BDAaaS framework by layer

Layer	**Component**	**Representative Examples**
BDSaaS	A wide range of domains	• Salesforce.com's Marketing Cloud • BrandsEye • Google's inventory management system
BDPaaS	Data Ingestion	• Batch: Sqoop • Stream: Kafka
	Data Storage	• DFS: Hadoop HDFS • NoSQL: MongoDB • RDBMS: Oracle
	Data Processing	• Batch: Hadoop MapReduce • Stream: Apache Storm
	Data Analysis	• the type of data analytics: descriptive, predictive and prescriptive • the type of approaches: statistics-based method and machine learning-based method
CIaaS	Computing	• Amazon EC2, Google Compute Engine, Azure Virtual Machines
	Storage	• Amazon S3, Google Cloud Storage, Azure Storage (Block Blob)
	Network	• Amazon VPC, Google VPC, Azure Virtual Network

Despite that BDAaaS is part of a broader 'as-a-Service' solutions such as SaaS/-PaaS/IaaS introduced earlier, unlike these common type of cloud services, BDAaaS

takes further to pay special attention to the technologies of big data analytics and its' application, which is more tailored to BDAAs.

2.6 Service Level Agreements

Service level agreements (SLAs) play an integral part for big data analytics applications in cloud since they serve as the foundation for the expected quality level of the service between providers and customers.

2.6.1 Definition of Service Level Agreements and Its Relevant Terms

2.6.1.1 Service Level Agreements

Multifarious definitions exist for the term service level agreement (SLA) [35, 114].

According to Marilly et al. [33], an SLA can be defined as follows: "a service level agreement is a contract between providers and customers, usually in measurable terms, what services providers will furnish and what penalties provider will pay if he cannot meet the committed goals". Hence, an SLA is a binding agreement that specifies what providers guarantees to deliver and can be offered to other providers or customers.

Gartner defines SLA as an agreement that sets the expectations between providers and customers and describes products or services to be delivered, the single point of contact for end users' problems and metrics by which the effectiveness of the process is monitored and approved [115].

IBM states that "a service level agreement is a contract between a provider and a customer that specifies the expectations for the level of service concerning availability, performance, and other measurable objectives. SLA records a common understanding about services, priorities, responsibilities, guarantees, and warranties between parties. SLA can also specify levels of availability, serviceability, performance, operation, or other attributes of the service" [116].

Information Technology Infrastructure Library (ITIL) has been an important factor in spreading SLAs. According to ITIL, SLA is an agreement between a provider and

customer. SLA describes the IT service, documents service level targets, and specifies the responsibilities of providers and customers [117].

Based on these diversified definitions, this thesis focuses on SLAs as negotiated "agreements" between different parties/entities. As "agreements", SLAs encapsulate a set of different aspects regarding the services provisioning. These refer to the agreed quality of service (QoS)– captured through different terms, the service level objectives that the service must guarantee in the form of constraints on QoS metrics, the responsibilities, and obligations of the parties, as well as the penalties in cases of non-compliance to the agreed terms. It specifies the levels of service that providers should provide to customers regarding objectives to attain different QoS aspects [35].

2.6.1.2 Service Level Objectives

Service Level Objectives (SLOs) refer to a set of formal expressions. These formal expressions have the well-known if...then structure. The antecedent (if) contains conditions and the consequent (then) contains actions. An action represents what a party has agreed to perform when the conditions are met [118].

SLOs are often quantitative and have related measurements. For customers who make informed decisions when choosing big data analytics as a service, it is best if the SLOs offered by each provider offering similar services can be easily compared.

2.6.1.3 SLA Metrics

SLA metrics represent measurement methods for the calculation of quality of service (QoS) values and define what services and guarantees provider will provide, which are often associated with a quantitative service level objectives (SLOs) [119, 120]. SLA metrics may be categorized as functional and non-functional. Functional properties cover aspects like the number of arguments and the semantics of operations. Non-functional properties define the service capabilities and robustness, covering terms regarding the QoS, security, and remedies for performance failures [121]. It is often true that a given SLO may have multiple different SLA metrics which can be used. It is essential that an SLA make it clear which metric(s) are being used for each quantitative SLO [122]].

2.6.1.4 Service Levels and Guarantees

Service levels and guarantees represent promises and guarantees with respect to graduated high/low ranges, e.g., average availability range [low: 95%, median: 97%, high: 99%,], so that it can be evaluated whether the measured metrics exceed, meet or fall below the defined service levels at a specific time point or in a certain validity period. They can be informally represented as if-then rules which might be chained in order to form graduations, complex policies and conditional guarantees, e.g., conditional rights and obligation with exceptions, violations and consequential actions: "If the average service availability during on month is below 95% then provider is obliged to pay a penalty of 20%. ".

2.6.2 The Evolution of SLA

Over the last thirty years, service level agreement (SLA) has undergone significant evolution driven by the advancement of distributed computing paradigm in order to adapt for changes and new challenges in different computing environments per requirements. We firstly give a brief description regarding the evolution of distributed computing paradigms, then explain our proposed SLA evolutionary stages over these years.

2.6.2.1 Distributed Computing Paradigm

According to the works in [123, 124, 125, 126, 127, 128, 129, 130], the distributed computing paradigm has evolved through a number of significant phases. Mainly, they are Internet Computing [131], Peer-to-Peer Computing [132], Cluster Computing [133], Grid Computing [134], Utility Computing [135], Cloud Computing (CC) [69] and Big Data (BD) [136].

The introduction of computer networks in the 1970s led to the development of distributed systems [137]. Then, the Internet (originally ARPAnet) was developed as a network between government research laboratories and participating departments of universities. Commercial Internet service providers began to emerge in the very late 1980s [138]. Up to this time, a few technologies emerged in the distributed systems. Peer-to-Peer network is one of the primary distributed systems with the purpose to

enable sharing of data, such as streaming audio or video [128]. In the 1980s, Cluster Computing has emerged, which is used for high-performance computing tasks. Another well-known distributed computing paradigm is Grid Computing that appears in the mid-1990s as an evolution of Cluster Computing [128]. In the 2000s, Utility Computing was proposed based on the idea of providing computing solutions in a very similar way as traditional real-world public utilities (such as electricity, water, gas, and telephone) [124]. Utility Computing was the first step towards pay-by-use philosophy. Around 2007, CC has emerged as a popular distributed computing paradigm [139]. CC is linked with Utility Computing based on the fact that CC is generally based on a pay-per-use model in which guarantees are offered using customized SLAs [140]. Recently, the further advancement of computer technologies and distributed processing paradigms have generated a new paradigm over the cloud at the forefront of BD. A representative example of such paradigm is MapReduce [141] programming model that is designed to work with distributed data-intensive processing big data analytics applications (BDAAs) in cloud. This has inspired an open source distributed computing framework called Apache Hadoop [94] and its ecosystem for cloud-hosted BDAAs.

2.6.2.2 SLA Evolutionary Stages

As an essential and efficient method of managing relationships between providers and customers and guaranteeing the level of service, we examined that SLA has been successfully used in all the aforementioned distributed computing environments over the last thirty years. Accordingly, SLA has experienced remarkable evolution as the distributed computing paradigm advances to cater to changes and new challenges in each distinct computing environment. Figure 2.3 proposes a pictorial representation of SLA evolution and lists some representative references in each particular evolutionary stage.

Concretely, the main stages in the SLA evolutionary roadmap include SLAs for Internet Computing [142, 143, 144], SLAs for Peer-to-Peer Computing [145, 146, 147], SLAs for Cluster Computing [148, 149, 150, 151], SLA for Grid Computing [152, 153, 154], SLAs for Utility Computing [155, 156, 157], SLAs for Cloud Computing [114, 118, 122, 158, 159, 160, 161, 162, 163] and SLAs for cloud-hosted BDAAs [164, 165, 166,

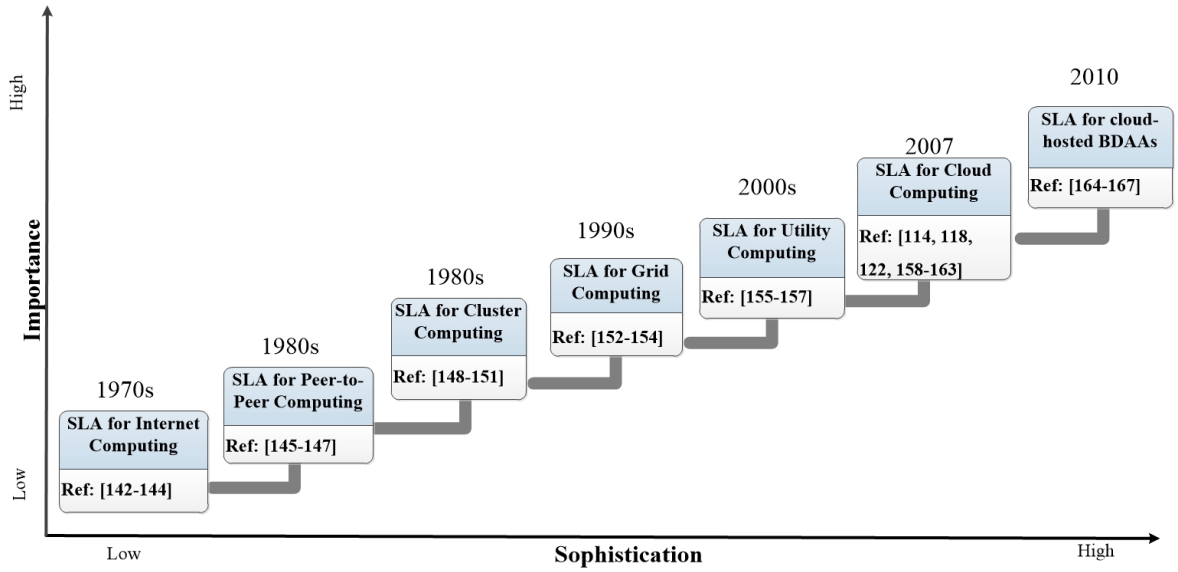


Figure 2.3: SLA evolution stages with representative references

167]. In this thesis, we are not going to detail each stage considering space. Instead, we give a brief explanation regarding SLAs for Internet Computing, SLAs for Cloud Computing and SLAs for cloud-hosted BDAAs.

Historically, SLAs have originated with internet service providers in the 1980s, which forms the first stage (i.e., SLAs for Internet Computing). Since the late 1980s, SLA's have been used by fixed-line telecom operators as part of their contracts with their corporate customers [168]. Various providers and customers need SLAs in the telecommunication marketplace. Hence, Internet service providers and telecoms will commonly include SLAs within the terms of their contracts with customers to define the level of service being sold in plain language terms. Further, in order to provide better practice advice, the Tele Management Forum had published the NGOSS SLA Management Handbook in 2001 which represents a milestone of SLA in its evolution [169, 170, 171]. Up to this time, NGOSS SLA Management Handbook is the most comprehensive and voluminous published collection regarding the management of SLAs with a focus on the Telecommunication Industry [172]. The drawbacks in this very early SLAs stage lie in the fact that the SLA metrics are limited to IP-based network performance measurements such as latency and packet loss, and the rigid specification

of the terms of SLAs, as it was not possible to adapt the values of SLA terms once they were deployed.

Since then, the SLA evolution continues from the stage of SLA for Peer-to-Peer computing to the stage of SLA for Utility Computing. Notably, the emergence of Grid Computing and Utility Computing triggered several important advancements in the specification of SLAs. This is because the openness and autonomy of grids and utility-oriented service provisioning model required specification formats that were not restricted to any organization or application domain's syntax or semantics.

Further, the rapid growth of the cloud market leading to the emergence of new services, new ways for service provisioning and new interaction and collaboration models both amongst cloud providers and service ecosystems drives the extensive exploitation of SLAs for cloud computing. The continuous technological advancement of distributed computing drives the application of SLA to the current stage of SLAs for cloud-hosted BDAAs. In this stage, the importance and sophistication of SLA increases than ever before. However, the research on this particular stage is far from mature.

It is worth mentioning that although CC is widely applied, it still has some restrictions. The fundamental restrictions lies in the correspondence between the cloud and the end devices. Such correspondence is not appropriate for a large set of cloud-based BDAAs such as the latency-sensitive applications (e.g.,disaster management, fire detection and firefighting, and content delivery applications etc.,). Hence, two new computing paradigms are proposed to address such issues. They are Fog Computing [173, 174] and Edge Computing [175, 176]:

- Fog Computing (FC): refers a novel architecture that extends the traditional CC architecture to the edge of the network. With fog, the processing of some application components (e.g., latency-sensitive ones) can take place at the edge of the network, while others (e.g., delay-tolerant and computational intensive components) can happen in the Cloud.

- Edge Computing (EC): refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services. The "edge" is denoted

as any computing and network resources along the path between data sources and Cloud data centers.

FC and EC are not substitutes for CC but powerful complements. Compared to CC, compute, storage, and network resources are still the building blocks of the Cloud in FC and EC environment. However, they extend CC by enabling provisioning resources and services outside the Cloud, at the edge of the network, closer to end devices or eventually, at locations stipulated by SLAs [174]. Therefore, SLA is still a fundamental element to manage the level of service for those applications deployed in FC and EC environment, especially for latency-sensitive ones [177, 178]. For FC and EC enabled BDAAs, SLA is often influenced by many aspects (e.g., energy usage, application characteristics, service cost, network status, data locations etc.,). [178]. Hence, SLA metrics in terms of these aspects should be one of their focusing points.

2.7 SLAs for Cloud-hosted BDAAs

This thesis focuses on the evolutionary stage of SLA for cloud-hosted BDAAs. Hence we will particularly analyze why SLAs are crucial and remain a complex challenging for cloud-hosted BDAAs, and how SLAs for cloud-hosted BDAAs are different from the previous stages in the following subsections.

2.7.1 Why SLAs are important for Cloud-hosted BDAAs

Although cloud computing (CC) offers lots of capacities to supply elastic, scalable, and cost-effective computational resources, cloud-hosted big data analytics applications (BDAAs) bring their own challenge regarding the provision of promised service and quality. Typically, domain-specific BDAAs (at BDSaaS layer) subscribed by customers will be efficiently processed at BDPaaS layer using advanced processing paradigm such as MapReduce [141] by leasing elastic resources from the underlying cloud datacenters (at CIaaS layer). In such an environment, services are provided strictly on an on-demand basis. Every kind of resource regarding services is offered in a pay-as-you-go model, which are charged by their actual usage. It is not hard to understand that the

management of SLAs for cloud-hosted BDAAs has become of paramount importance due to the following aspects.

- Customer commitments: Clearly defined promises regarding big data analytics service and its service level reduce the chances of disappointing customers and compromising the fulfillment of the value of BDAAs. These promises also help to stay focused on serving customers' requirements and assuring SLA guarantee.

- Increased interaction between providers and customers: Providers' and customers' staff are more involved in the loop than in traditional cloud providers offering. In complex cloud-hosted BDAAs deployment scenarios, there is not merely a single business entity responsible for the service delivery, but multiple different layers, business actors and entities with contrasting and competing requirements (i.e., minimum cost VS. minimum response time) interact to provide analytics services to customers and finally end users.

- Strong dependency relationships across layers: Notably, there are multiple SLAs with different properties and attributes across layers for a cloud-hosted BDAA. These SLAs have a strong dependency relationship with each upstream and downstream layer. The variety of one SLA metric such as the number of VMs (at CIaaS layer) might affect the other one such as the response time of data processing (at BDPaaS layer), which will result in the breach of the application-level SLAs to customers.

- Key performance indicators for customers: By having "agreed" service delivered and SLA guaranteed, the confidence of customers to gain valuable information and extract insight by using the applications could be increased. In the light of this, it would be easy for customers to make an evidence-based decision.

- Key performance indicators for providers: SLA drive internal processes of providers by setting a clear, measurable standard of their service. Consequently, service level objectives become clearer and easier to measure, and providers could achieve their competitive advantages in the market of analytics service or solutions.

- The price of non-conformance: If an SLA has penalties, non-performance can be costly. Therefore, by having penalties defined, customers understand that providers truly believe in their ability to achieve the performance levels. It makes the relationship clear and positive.

To better understand why we should pay attention to SLA-specific management for cloud-hosted BDAAs, we take social-network-driven stock recommendation as an instance [179, 180]. This is a domain-specific cloud-hosted BDAA deployed on Amazon AWS datacenter. Figure 2.4 presents the architecture of this application, which is an instantiation of the aforementioned layered-based architecture of cloud-hosted BDAAs. It is observed that the architecture of this cloud-hosted BDAA consists of multiple and heterogeneous components to process both batch- and stream-based data. The application integrates the functions of streaming free text (via Twitter API) and batching recorded tweets (through Twitter Firehose) that are contained in Amazon S3 storage buckets [93]. It consists of the following key components:

- Apache Kafka [97] is served as a data ingestion framework featured by high throughput.
- Apache Storm [103] is an advanced stream-oriented data processing engine.
- Apache Hadoop [94] represents a batch-oriented data processing framework for historical tweets.
- Apache Mahout [181] implements a set of machine learning algorithms (e.g., Naive Bayes) for grouping and classifying tweets, and runs on top of Apache Hadoop runtime environment.
- Yahoo Scalable Advanced Massive Online Analysis (SAMOA) [182] is deployed as a data analysis framework for grouping and classification of tweets pertinent to a specific stock.
- Hadoop Distributed File System (HDFS) [183] stores the final result of both batch and stream processing.

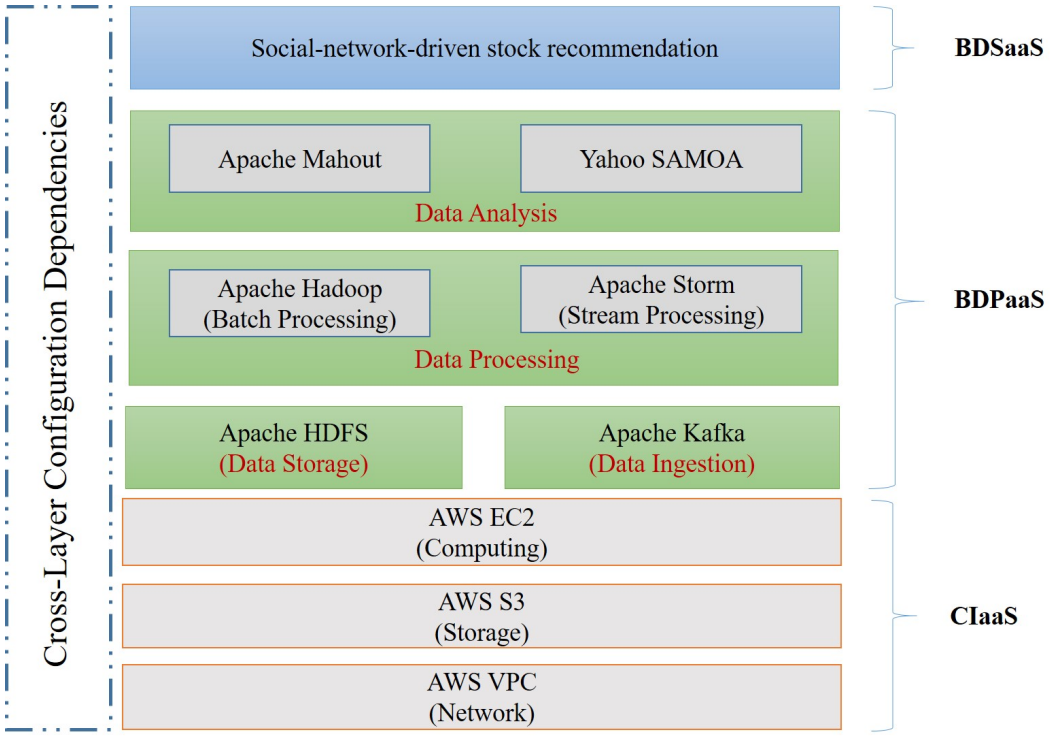


Figure 2.4: A stock recommendation BDAA hosted in AWS Datacenter

In the context of this application, it is not hard to understand that SLA management is very important. Concretely, this application uses live Twitter sentiments and stock data to analyze the stock market movement and correlation in order to recommend buying and selling points and promising combinations of stocks to investors in real time. In this BDAA, data analysis results should be fed back to the investors through highly efficient processing of queries, so that investors can decide just-in-time when and which stocks they should purchase or sell. Such BDAA cannot tolerate delays (unsatisfactory SLA) in the data processing (the fast movement of the stock market, the impact of politics, economy and many other factors) required for fast evidence-based decision making. Therefore, data analysis delays (in this case, SLA is breached) could lead to significant monetary losses. Although we consider a stock recommendation BDAA here, the importance of SLA applies to other types of applications as well such as inventory management, healthcare, topic detection, sentiment analysis, smart cities and so on.

2.7.2 Why are SLAs for BDAAs in Cloud sophisticated?

Further considering the above instance, the management of SLAs in this context is extremely challenging due to the complexities imposed by this application in nature. For instance, the application spans heterogeneous and distributed software frameworks across multiple layers, which considerably impacts the allocation and configuration of datacenter resources to accommodate dynamic big data workloads and guarantee analytic results within SLA constraints. Each component of this application has different SLA requirements. For example, the application-level SLA constraints (i.e., minimize query processing, minimize data analysis delays, maximize application and data availability), platform-level SLA constraints (i.e., maximize Kafka ingestion speed, maximize HDFS I/O throughout, maximize MapReduce tasks and stream processing, minimize Mahout analysis delay), while optimizing the resources utilization of cloud datacenter in terms of CPU, network, storage and energy. It is very hard to tell how these SLAs could be defined coherently across layers and what hidden patterns among multiple configurations of this application posses.

This level of complexity in terms of SLA management also applies to other kinds of BDAAs although we only take stock recommendation application as an example.

2.7.3 How is SLAs for BDAAs in Cloud different from the previous evolutionary stages?

Compared with its previous stage (i.e., SLAs for Cloud Computing), the stage of SLAs for cloud-hosted BDAAs demonstrates remarkable disparities in some aspects. For example:

- Conceptual SLA model: BDAAs are transforming corporate decision-making, pushing organizations to become more agile and responsive. The new service features, the cutting-edge big data technologies, new interaction and collaboration mechanisms between actors and higher service quality requirements lead to different conceptual SLA model in cloud-hosted BDAAs scenario.

- SLA metrics: The discrepancies of SLA metrics occur between cloud computing environment and BDAAs environment as the nature and type of resources being provided and delivered are entirely different. Although some researchers propose SLA metrics such as availability, scalability, response time, and so on that could be applied for BDAAs in cloud, it is still vital to investigate and classify SLA metrics for BDAAs in cloud in a new way reflecting characteristics of BDAAs.

- Techniques used to address SLA management: The rapid technological development brings changes to the way that SLA is managed and guaranteed for cloud-hosted BDAAs. For example, scheduling is a favorite technique to address SLA management problem in prior stages. However, they cannot be directly used in cloud-hosted BDAAs scenarios because the optimization objectives, mathematical formulas, and system architect are different. Hence, new algorithms or solutions should be designed and developed to deal with the scheduling changeling for cloud-hosted BDAAs.

- Layers: According to the aforementioned big data analytics as a service framework, service layers themselves and technologies in these layers bring remarkable disparity between general applications in cloud computing environment and cloud-hosted BDAAs, which transforms the way we understand and address SLA management. Concretely, different algorithms, tools, or methods should be considered and applied while understanding and respecting such disparities.

2.8 Summary

In this Chapter, we first introduce some fundamental terms and concepts of big data (BD) and cloud computing (CC). The relationship between BD and CC has been elaborated. Next, we provide essential information regarding big data analytics applications (BDAAs) in cloud and present a layer-based big data analytics as a service framework. Then, critical information about SLAs and its relevant terms have been given. We discuss how SLAs evolve as the distributed computing paradigms advance over the last thirty years. Moreover, we analyze why SLAs are important, why SLAs is

a challenging problem and how SLAs are different from the previous evolutional stages in the context of BDAAs in cloud.

In the next chapter, we will use the methodology of systematic literature review to conduct an in-depth survey regarding SLA management for big data analytics applications in cloud in response to *RQ1*.

Chapter 3

Systematic Literature Review on SLA Management for Cloud-hosted BDAAS

In this chapter, we will address the first research question defined as *RQ1: What is the landscape of the extant research on management of service level agreements (SLAs) for big data analytics applications (BDAAs) in cloud and how to classify them from different perspectives?*

To this end, a research methodology, i.e., a systematic literature review has been applied to the published papers in this field. The results contribute to the fundamentals of engineering SLAs for cloud-hosted BDAAs and management of SLAs, motivating further research and industrial-oriented solutions. This systematic literature review not only enhances the knowledge for the general readers and researchers but also identifies open challenges and gaps that need to be addressed in future research directions.

3.1 Positioning versus Existing Surveys

A few previous studies focused on the literature review of SLA management in the context of cloud environment. Mohamadi et al. [184] conducted a very preliminary review of SLA management approaches and compared them with respect to improved parameters, implementation/simulation and its environment, and workload/application. Faniyi et al. [185] surveyed the research landscape of SLA-based cloud systematically with the focus on specific phase of SLA life cycle (i.e. resource allocation phase) and outlining consequences on the others. From cloud service provider perspective with small to medium-sized enterprise level. Hussain et al. [186] presented a comprehensive

overview of existing approaches of SLA management in clouds and highlighted the features and limitations of these approaches to tackle the issue of creating a viable SLAs in cloud computing from the viewpoint of service provider's. While from another perspective, Whaiduzzaman et al. [187] focused on SLA-based service provisioning techniques and methods that assist in evaluating cloud services provisioned with regards to user-specific requirements and cost.

With the recent advancements in computing, internet of thing (IoT) has been introduced as an emerging and promising technology. Thus, Saad Mubeen et al. [188] conducted a survey to investigate the existing research on SLAs management for IoT-based applications in clouds. This survey used a systematic mapping study for the purpose of identifying the results of the published research works that are related to SLA management in IoT context.

Existing surveys either focus on SLA management in cloud environment or IoT environment, which are not specific and sufficient for SLA management in the context of cloud-hosted BDAAs. To the best of our knowledge, there is only one survey work has been done on SLA management for BDAAs in clouds. Sahal et al. [189] conducted a preliminary survey and divided SLA management into two types, which are SLA management for cloud computing and SLA management for cloud-hosted BDAAs. Regarding to latter type, SLA management approaches are categorized into two groups comprising MapReduce scheduler and cloud Layer, which we argue that this categorization is oversimplified and fails to give a holistic landscape towards SLA management for BDAAs in clouds.

Therefore, we need a systematic literature review (SLR) on SLA-specific management for BDAAs that covers up SLA model for this type of application with respect to its requirements across cloud computing stack. Our work in this Chapter is the positioning attempt to conduct this in-depth SLR with build-up thematic taxonomy covering six core dimensions including actors, service layers, techniques, cloud service and deployment models, SLA metrics and conceptualization. This work not only provides a comprehensive review of the state-of-the-art landscape on SLA-specific management for BDAAs in the cloud, but also insights into understanding the research themes/patterns in this field.

3.2 The Application of SLR Methodology on SLA Management for Cloud-hosted Big Data Analytics Applications

Our study in this Chapter uses a systematic literature review (SLR) methodology proposed by Kitchenham et al. [190] to review the landscape of SLA-specific management for cloud-hosted BDAAs. SLR is a scientific, objective, transparent and reproducible method of reviewing extant literature to answer and deduce particular research question(s) in such a way that is unprejudiced. Adopting this research methodology provides a systematic process aimed at identifying the state-of-the-art in the research study field. Our SLR consists of three main phrases as shown in Figure 3.1.

The first phase is planning for review, where we define our research questions that will be considered and tackled in this study and determine the strategy of search being used for primary studies with lists of inclusion and exclusion criteria. After the executing search query on the selected database sources, the relevant papers obtained will be input for the next phase. The second phase is about conducting the review by applying the selection mechanism including primary selection (exclusion and inclusion criteria) on the obtained papers to get all relevant papers, and the secondary selection for performing a further evaluation for each remaining paper to get the most relevant papers. In the final phase, we further analyze the remaining papers to report a thematic taxonomy of SLA management for BDAAs in clouds and review these papers based on this taxonomy.

3.2.1 Systematic literature Review Questions

Our SLA aimed at capturing the research landscape in SLA management for BDAAs in clouds by addressing the following systematic literature review questions (*SLRQ*):

- *SLRQ1*: What are the actors involved in making conversations and engineering SLAs in the context of cloud-hosted BDAAs?
- *SLRQ2*: What is the status of addressing SLA management for cloud-hosted BDAAs from different service layers (i.e. BDSaaS, BDPaaS and CIaaS) and cloud

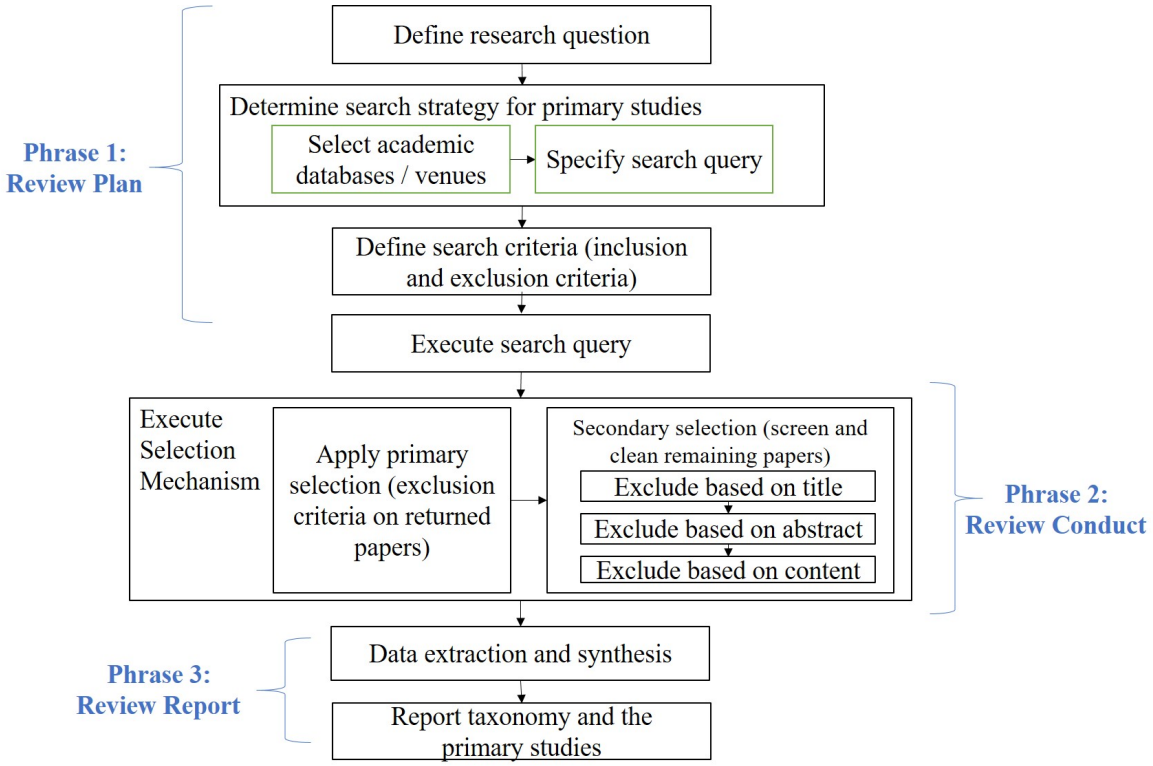


Figure 3.1: Systematic literature methodology applied on the survey of SLA management for cloud-hosted BDAA

deployment models (i.e. private, public, hybrid and community cloud)?

- *SLRQ3*:What are the techniques applied to address SLA management for cloud-hosted BDAAs?
- *SLRQ4*: What SLA metrics are of interest to stakeholders and being discussed in the context of cloud-hosted BDAAs?
- *SLRQ5*: To what extent the SLA model for cloud-hosted BDAAs is conceptualized?

3.2.2 Search Strategy for Primary Studies

3.2.2.1 Selection of Academic Databases

To search for research publications in the areas of computer science, there are well-known databases that are being used as primary sources for these publications [190]. For our study, the academic databases selected are shown in Table 3.1. These databases provide advanced search options with a set of Boolean functions to make concise search based on certain fields such as abstract, title and keywords, which return the most relevant results in comparison to search all fields.

Table 3.1: Academic database sources

Source	**URL**
IEEE Explore Digital Library	http://ieeexplore.ieee.org
ACM Digital Library	http://portal.acm.org
Springer	http://springerlink.com
Science Direct	http://sciencedirect.com
Web of Science	http://webofknowledge.com
Google Scholar	http://scholar.google.com

3.2.2.2 Search String

In this step, we construct our search string that will be executed over the aforementioned databases to search relevant publications. To provide comprehensive coverage of the relevant research works in the literature and state-of-the-art studies, we need to select keywords carefully. Thus, we consider the terms “service level agreement”, “Quality of Service”, “big data” , “big data analytics”, “big data analysis” and “Big Data Analytical Application” as the primary keywords along with a range of related abbreviations, plural or synonyms, namely “service level agreements”, “service-level agreements”, “SLA”, “SLAs”, “SLM”, “SLA management”, “QoS”, “BDA” and “BDAA”. Since a typical BDAA comprises data ingestion, data storage, data processing and data analysis framework, we also consider “MapReduce”, “batch processing”, “batch computing”, “stream processing”, “stream computing”, “data ingestion” and “NoSQL”

Table 3.2: List of Inclusion Criteria

Criterion	**Description**
The type of study is peer-reviewed	We chose peer-reviewed publications including conference/workshop and journal papers, and peer-reviewed book chapters.
The writing language for the study is English	We restrain the language to English because some databases such as Springer returns publications in another language like German
The publication year for the study is published from 2010 to 2018	We search all publications in accordance to our search string that have been published in the databases between 2010 and 2018.

as additional keywords. To join the primary keywords and the additional keywords with their synonyms in the search string, Boolean functions (AND and OR) are used. Moreover, to make sure that our search string returns as many relevant studies as possible in the selected databases, we conduct several tests. As a result, we select the following search string:

(“service level agreement” OR “service-level agreement” OR “SLA”) OR (“service level agreements” OR “SLAs”) OR (“SLA management”) OR (“service level management” OR “SLM”) OR (“SLA conformance”) OR (“quality of service” OR “QoS”)

AND

((“big data”) OR (“big data analytics”) OR (“big data analysis”) OR (“Big Data Analytical Application” OR “BDAA”) OR (“Big Data Analytical Applications” OR “BDAAs”) OR (“big data analytics” OR “BDA”) OR “MapReduce” OR (“batch processing”) OR (“batch computing”) OR (“stream processing”) OR (“stream computing”) OR “NoSQL” OR “ingestion”))

3.2.3 Search Criteria

To evaluate the publications that will be obtained after applying the search string in next phase, we need to define the search criteria that being applied. Table 3.2 and 3.3 show inclusion and exclusion criteria that are performed in our SLR.

Table 3.3: List of Exclusion Criteria

Criterion	**Description**
The focus of study is not SLA management	We consider the publications in the field of SLA management for cloud-hosted BDAAs.
Abstracts and publications that are not pass through refereeing process	We exclude the study with only abstract, not in the form of full paper and is not peer reviewed.
Duplicate publication	We remove duplication for the same study found in different databases

3.2.4 Selection Mechanism

After executing the search string and applying inclusion and exclusion criteria on the obtained relevant publications in the field of SLA management for cloud-hosted BDAAs, we get the initial result of 1098 papers. Then, we conduct the stringent selection (secondary selection) on them based their titles, abstracts and contents using the following rule. We strictly select publications that consider SLA management specifically in the context of BDAAs in clouds and exclude those publications that only discuss SLAs in general cloud computing environment. This is because, our focus in this thesis is given to the evolutionary stage of SLAs for cloud-hosted BDAAs rather than the evolutionary stage of SLAs for Cloud Computing. Moreover, SLA management in Cloud Computing is a well-explored area with lots of papers. However, SLA management for cloud-hosted BDAAs is still a young research area, which demands more necessity and urgencies study on. At the end of this phase, we get 109 papers that will be systematically reviewed in this study.

3.2.5 Data Extraction and Synthesis

The quality assessment strategy used in this study is subjective analysis, where we evaluated the collected papers from primary studies to assess their relevance to the landscape of the survey. From this analysis, the thematic taxonomy of SLA management for BDAAs in clouds has been emerged (see Figure 3.2). The following are the detailed descriptions for the proposed taxonomy elements:

- Actors − This element considers the different actors (service providers, consumers and cloud end users) involved in cloud-hosted BDAAs. Service providers provide

the consumers resources that can be provisioned and metered on demand including big data platform resources and cloud infrastructure. In particular, it could further classified into BDSaaS, BDPaaS and CIaaS providers. These providers care more about the efficient resource utilization, energy efficiency, profit maximization, cost reduction, and performance enhancement. The service consumers are those actors that utilize services offered by service providers and are liable for their resource consumption's, where they are more concerned about the budget, pricing, and customer satisfaction. The cloud end users (or for short end users) are those actors that use the applications or services offered by service customers. They are more interested in QoS and SLA constraints such as service quality, performance and response time.

- Service Layers − This element examines SLA management from various levels of abstraction including CIaaS, BDPaaS and BDSaas. At CIaaS layer, SLA management takes care of guaranteeing SLA requirements on virtualized resources such as VMs, storage or network. At BDPaaS layer, such management guarantee SLA requirements for different big data frameworks including data ingestion, data processing, data analysis and data storage), and at BDSaaS, such management cares about guaranteeing user-specific application requirements.

- Techniques − In the context of cloud-hosted BDAAs, one or more techniques (such as optimization-based, scheduling-based and simulation-based) can be included into SLA management to guarantee SLA requirements at specific or across service layers. A technique is a method that aims to address SLA management for cloud-hosted BDAAs.

- Cloud Deployment Models − When proposing a new SLA management, the activities like implementation, deployment, validation and evaluation come to the picture in order to access the validity and practicality of such management in cloud infrastructure. Thus, these activities need to be carried-out in private, public, community or/and hybrid cloud deployment model, where each one of them has its own requirements and challenges.

- SLA Metrics – This element considers SLA items that are defined as quantitative targets in the contract and they must be maintained by the service provided. Measuring these items/metrics (such as cost, deadline and performance) is critical for SLA management to avoid any SLA violation.

- Conceptualization – This element examines SLA management from conceptualization perspective instead of concrete techniques. It mainly consists of designing conceptual SLA models that offer researchers a fundamental and clear way to describe actors, activities and entities involved in a cloud-hosted BDAA scenario and understand the context of SLA including the conversation and relationship between providers and customers.

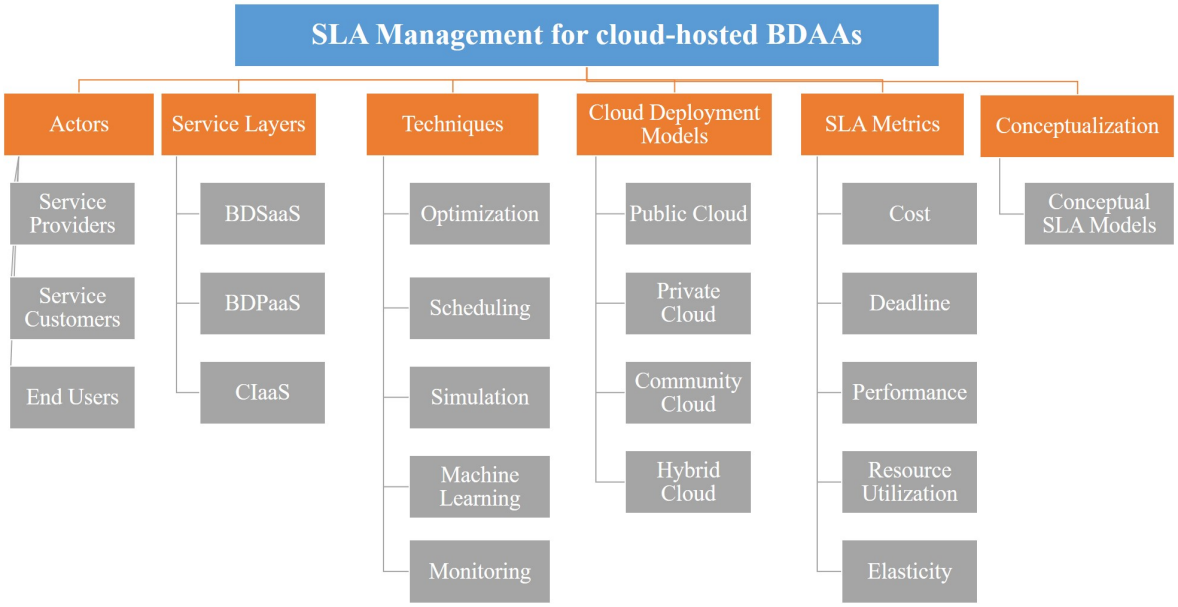


Figure 3.2: Taxonomy of SLA management for cloud-hosted BDAAs

3.3 Review Results in Thematic Taxonomy of SLA Management for cloud-hosted BDAAs

3.3.1 Actors (In response to *SLRQ1*)

3.3.1.1 Providers

In terms of providers' profit maximization, the authors [191] provide an SLA-based PaaS architecture that can support cloud-hosted BDAAs. In their paper, a disperse optimization policy is proposed, which aims at maximizing providers' profit and considers to pay penalties incurred when SLA are unsatisfied. Then, the proposed optimization policy is applied to cloud-hosted BDAAs (e.g., MapReduce applications). In paper [165], the authors designed and implemented automated and elastic resource scheduling algorithms with the objective of profit optimization. Their algorithms can deliver BDAAs to users and optimize profits of platforms while guaranteeing SLAs for query requests in terms of deadlines and budgets and allowing prompt responses with manageable financial costs.

Unlike the above works, paper [192] focuses on the optimization of energy consumption from providers' perspective. The authors take into account sharing MapReduce-based applications in an environment of Hadoop YARN and introduce an SLA-driven energy-saving scheduling algorithm for them [193]. Job profiling is performed to capture the characteristics of performance for diverse stages of a MapReduce-based BDAA. The obtained characteristics of performance will be considered as input to resource provisioning phrase with the purpose of guaranteeing application's SLA such as the completion deadlines. Their experiments demonstrate that their approach enhances the conformance of SLA in terms of reduced energy consumption and resource expenditure.

3.3.1.2 Customers

The authors in [194] focused on their study on cloud-hosted databases from the customer perspective and addressed the challenge of SLA-driven provisioning and cost management for them. In their paper, a comprehensive framework is proposed, which

can flexibly and dynamically provisioning cloud-hosted database of BDAAs. According to application-defined policies, their proposed framework can satisfy SLAs in terms of performance requirements, avoid penalties when SLA violations occur and control expenses when allocating computing resources.

3.3.1.3 End Users

The authors in [195] proposed an improved resource revenue optimization model. The model defines the constraint mechanism that describes quality of service (QoS) problems. They sliced the requests of end users, modeled the process of requesting service, evaluated the time of response and processing, and allocated resources based on the specified objective function while considering end users' requirements for QoS in this model. They designed a parallel and distributed algorithm based on the working mechanism of MapReduce to solve their proposed model while guaranteeing end users' needs on QoS as much as possible.

3.3.2 Service Layers (In response to *SLRQ2*)

Figure 3.3 presents the statistics of SLA management works in the reviewed papers by the service abstraction level and their breakdown in each layer.

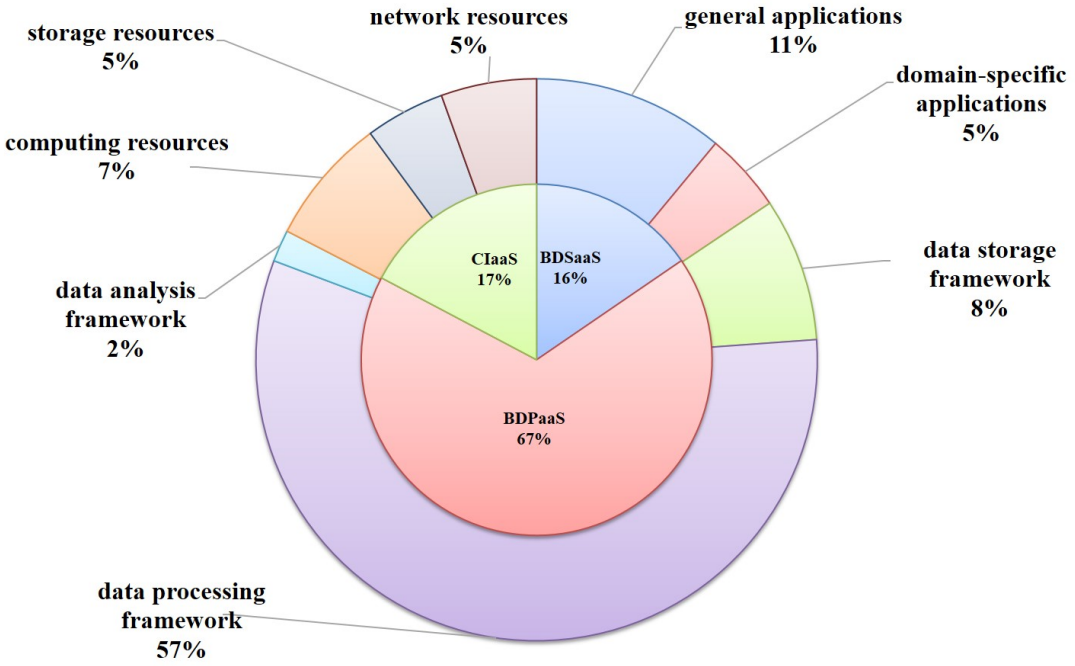


Figure 3.3: Statistics of the reviewed papers by layer and their breakdown

It is observed that most of the reviewed papers (67%) fall into the BDPaaS sector. This demonstrates that BDPaaS is the core part of BDAaaS and attracts more interest from researchers. When drilling down into the BDPaaS layer, we found that the top-ranked framework at this layer is data processing with a percentage of 57%. This is because that distributed data processing technologies such as MapReduce receive lots of attention in academia since 2010. Also, it is seen that the data storage framework is ranked secondly, occupying 8%. This indicates that representative data storage technologies such as NoSQL are of increasing interest by researchers in recent years.

Moreover, from the BDSaaS sector, there are 11% reviewed papers discussing SLA management for general BDAAs, while 5% reviewed papers providing domain-specific BDAAs. Interesting, it is further noted that among these domain-specific BDAAs, healthcare application is of the most interest for researchers with four reviewed papers [165, 196, 197, 198] in total and only one reviewed paper takes banking application as the case study [199]. Besides, in CIaaS sector, it is seen that the computing, storage, and network components share the balanced percentage, which means they receive even

attention in academia.

Additionally, we present the works of SLA management for cloud-hosted BDAAs by layers with the corresponding references in Table 3.4. It has been seen that the quantity of publications regarding batch processing is much higher than that regarding stream processing. The reason is that typical batch processing paradigm like MapReduce featured by its automatic parallelization and distribution, fault tolerance and simplicity becomes ubiquitous programming framework to parallelize the processing of large dataset, which gained significant interest both industry and academia since its emergence in 2007. However, stream processing such as Spark or Storm has been earning improving attention in the last years due to the emerging need for supporting a real-time or near-real-time processing task. The representative works in each layer will be discussed in the following subsections.

3.3.2.1 BDSaaS layer

General applications: The authors [203] addressed the challenge of situations where numerous job instances in BDAAs should be concurrently deployed at runtime. They introduced DepWare that is a specialized middleware capable of offering an autonomic deployment decision making. They then designed DepPolicy that is a novel language to specify fundamental deployment information. Moreover, an algorithm of deployment decision making is proposed to achieve the optimum deployment for each job instance. Experiments shows that their algorithm of deployment decision making can simultaneously make diverse decisions of deployment at runtime for different job instances. Meanwhile, optimal overall utility is achieved, all given constraints (e.g., cost limit) is satisfied and SLAs (e.g., feasibility, functional correctness, performance, and scalability) is guaranteed.

Domain-specific applications: The authors in [198] focus their work on healthcare BDAA where missing any SLA can generate significant influence on the data analysis of emergency patient thanks to the disease severity. They proposed a computing model for SLA-based healthcare BDAA and implemented a single API to manipulate and analyze both stream-based and batch-based data over Spark platform. They then presented a probabilistic method based on parallel semi-Naive Bayes (PSNB) and

Table 3.4: Classification of the reviewed papers on SLA Management for BDAAs in clouds by layers

Service Layer (# of papers)	**Category (# of papers)**	**Sub-category and References**
BDSaaS (17)	Applications (17)	• General: [164, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210] • Domain-specific: [211, 196, 197, 198, 199]
BDPaaS	Data Processing Framework (63)	• Batch processing: [212, 213, 165, 192, 191, 195, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259] • Stream processing: [197, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269]
	Data Storage Framework (9)	[194, 270, 271, 272, 273, 274, 275, 276, 277]
	Data Analysis Framework (2)	[278, 197]
CIaaS	Computing resources (8)	[277, 279, 280, 281, 282, 283, 284, 285]
	Storage resources (6)	[286, 287, 288, 289, 290, 291]
	Network resources (6)	[292, 293, 294, 295, 296, 297]

designed a modified conjunctive attribute algorithm for dimensionality reduction to improve the accuracy. For those jobs with high priority, they proposed an adaptive job scheduling algorithm to optimize their execution time that satisfies SLA. Experiments results show that their proposed model for SLA-based healthcare BDAA outperforms extant parallel processing models. Also, their proposed PSNB-based approach enhances accuracy compared to the original Naive Bayes algorithm. Differently, the authors [199] focus on how to schedule BDAA workflows in both single cloud and federated inter-cloud environments. A workflow consists multiple tasks that need storage, computing and bandwidth resources to transmit and process data. The resources in the workflow need satisfying SLA requirements (e.g., optimizing time to meet deadlines, optimizing cost and managing budgets). A cloud or inter-cloud provider could provide resources for executing tasks in the workflow according to specified SLAs criteria. A case study of banking application demonstrates that single or federated cloud resources are very necessary in terms of executing BDAA workflows.

3.3.2.2 BDPaaS layer

Data Storage Framework: Sakr et. al. [274] considers cost management and SLA-based provisioning for cloud-hosted NoSQL databases. In their paper, they proposed an end-to-end framework that is represented as middleware residing between the cloud-hosted databases and consumer applications. The proposed framework aims to flexibly and dynamically provisioning one database function in BDAAs while satisfying their SLA performance requirements (e.g., variability, scalability, elasticity, and performance) according to application-defined policies and avoiding the monetary cost when SLA violations happen as well as controlling the expenses when allocating computing resources. In the context of data security in NoSQL database, Crypt-NoSQL [270] is the first prototype that can encrypte data and execute queries on NoSQL databases while providing high performance. The authors in this paper proposed three different types of models for Crypt-NoSQL and evaluated its performance using Yahoo! Cloud Service Benchmark. Their experiments demonstrate that Crypt-NoSQL is able to efficiently execute queries while guaranteeing SLA requirements (e.g., scalability, high performance). Moreover, they proposed guidance for providers to establish Crypt-

NoSQL in the form of a cloud service and set up pertinent SLA conventions.

Data Processing Framework: On the one hand, some papers focus on batch-based MapReduce jobs in cloud. For example, the authors [221] aim at minimizing SLA metrics (e.g., response time) and keeping deadlines set in the pSLA (platform-level SLAs) in this context. First, they developed a so-called grey-box model that can accurately obtain the characteristics of MapReduce behavior. They then proposed a control theory-based framework to satisfy the objectives of SLA. A feed-forward controller was designed and implemented to assure constraints such as service time and improve control response time. The experiments illustrate that the controller is valid in meeting the specified deadlines in the SLAs. Lim et al. [247] propose a novel MapReduce resource manager using constraint programming-based method. In this paper, each MapReduce job is featured by a set of metrics (e.g., the time of earliest start, the time of execution, and deadline) specified in an SLA document. The authors evaluated the performance of their resource manager through an open and discrete event-based simulator where a stream of jobs arrive at intervals. The experiments show that the resource manager can achieve good performance in matchmaking and scheduling MapReduce jobs and give insights into the behavior and performance of system. On the other hand, some papers focus on stream-based jobs. For example, Rafael et al. [265] take into account simultaneously executing stream workload over shared cloud infrastructures where each stream is characterized by specific quality of service (QoS) objectives (e.g., throughput, latency) specified in an SLA. They consider classifying customers who submit streams workload into three different classes (Gold/Silver/Bronze). Each class differentiates by a unique penalty and revenue from providers' side. Their proposed profit model can consider both the cost of provisioning and penalties when the violations of SLA occur. Experiments show that their approach can apply the enforcement of QoS for each application. Paper [266] discusses provisioning resources for stream-based jobs at a granularity of VMs level at runtime. They proposed a novel method to provisioning resources in a cost-efficient way while optimizing the resource usage of VMs (SLA metrics at CIaaS layer). Moreover, their method is integrated into the Vienna ecosystem at runtime environment for scalable stream processing. The evaluation shows that their method achieves better conformance of SLA by up to 25% and the operation cost

reduction up to 36% compared to the extant threshold-based method.

Data Analysis Framework: The authors [278] developed an extensive model for predictive analysis regarding performance and cost in cloud. They collected data of resource consumption and placed them in readiness state to enable fast analysis. They stored time series data and various kinds of data regarding performance and events of BDAAs in a layered object store, which can provide the abilities of fast retrieving and pattern analysis. Meanwhile, the authors took into account the data aggregations regarding the interrelation between performance and cost as well as their dynamic tendency over time. Hence, through the application of real-time predictive analysis techniques, the framework achieves an accurate prediction of the current status (i.e., cost and performance) and prospective status. This provides effective support for providers to make decision based on resource configuration regarding the guarantee of SLA requirements. With regards to analytical capability on prediction accuracy, Lekha et al. [197] focus on developing a real-time stream-based system for the prediction of health status of patients. The system is implemented and deployed on a cloud-hosted Spark platform which leverages the power of multiple machine learning algorithms. In this scalable system, first, the health information tweeted by users are captured. Then, the proposed system can receive the same health information in real time. Next, the system preprocesses and extracts valid health information from those unstructured stream data and utilizes machine learning algorithms to forecast the health status of users with the purpose of maximizing the accuracy of prediction. By the virtue of the availability of high-quality training datasets of healthcare and the computing power of steam processing of Spark, the process of analyzing huge healthcare samples by applying machine learning techniques becomes significantly more efficient than ever, resulting in enhanced prediction accuracy.

3.3.2.3 CIaaS layer

In terms of cloud storage, paper [286] addressed various requirements of customers regarding cloud storage. They first defined a set of realistic and concrete SLA elements. Then, they designed the short-secret-sharing cloud storage system that applies the defined SLA elements and provides customers with a protected and steady storage

service in cloud. Their proposed system can capture applicable parameters to offer customers with their wanted services while respecting SLAs (e.g., minimal costs). The authors in [288] focus on the scheduling policies of volume request that can shorten the violations of SLA in terms of I/O throughput in cloud storage systems. To this end, they propose various SLA-driven scheduling policies that consider both I/O throughput and available capacity of backend nodes. The designed scheduling policies can considerably reduce monetary cost of cloud storage from providers' perspective.

Unlike the above works, Yassine et al. [294] discussed the challenge of transferring multimedia big data over cloud data centers that are geographically distributed. Since the multimedia volume increases, there is an increasing demand to transfer large datasets across data centers. Therefore, the surplus bandwidth that occurs at different times and for different period in backbone network turns to be inadequate to meet speedily increasing demand for transferring multimedia big data. In this paper, they designed multi-rate Bandwidth-on-Demand (BoD) service to communicate among geo-distributed cloud datacenters. They also developed a scheduling algorithm that is employed by a BoD broker, which considers transferring multimedia big data requests that are featured by various deadlines.

3.3.3 Cloud Deployment Models (In response to *SLRQ2*)

In the reviewed papers, researchers select a particular type of cloud deployment model (private/public/community/hybrid) to carry out various activities such as implementing the proposed framework, deploying prototype or tools, evaluate approaches or simulate testing environments. Figure 3.4 shows the distribution of cloud deployment models in the reviewed papers.

It is worth noting that a significant fraction of the reviewed papers only mention cloud service in general without explicitly telling the cloud deployment model they used. In this case, we label it as "unknown cloud deployment model". Apart from the sector of "unknown cloud deployment model", it is observed from Figure 3.4 that public cloud is accredited as the principal cloud deployment model. This finding is in line with our understanding that the public cloud is the most common and well-known deployment

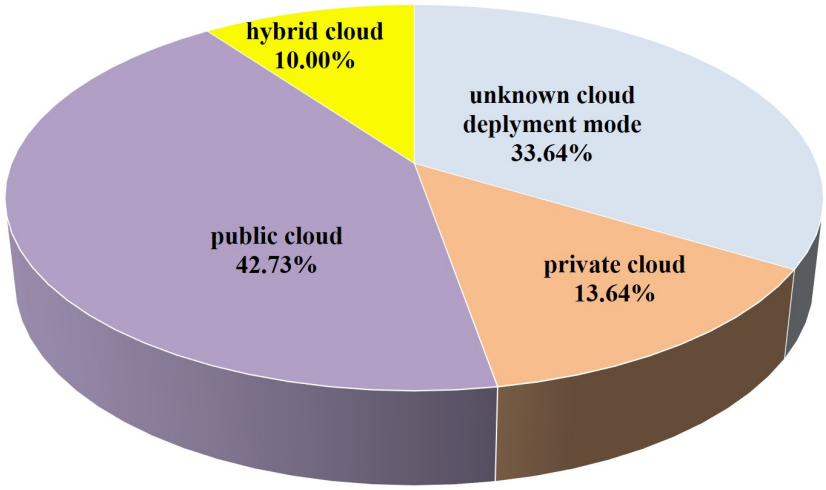


Figure 3.4: Statistics of the reviewed papers by cloud deployment models

model in comparison with the others. Accordingly, researchers are prone to choose public cloud to deploy their proposed prototype, applications or tools and evaluate their proposed techniques. The second and third-ranked sectors are private cloud and hybrid cloud, occupying 14% and 10% respectively. Interestingly, community cloud is not used in the reviewed papers.

Regarding public cloud, we further investigate what specific public cloud platforms were selected. Figure 3.5 gives an apparent breakdown of the public cloud. It is found that some papers fail to state what specific public cloud platform was used. Hence we mark them as "unknown public cloud platform" in Figure 3.5. It is seen that Amazon EC2 is the preferable public cloud platform, occupying 29%. Comparatively, only 2% of the reviewed papers use Microsoft Azure as their deployment platform.

Next, Figure 3.6 presents the breakdown of private cloud. Similarly, some papers fail to state what specific private cloud platform was used. Hence we label them as "unknown private cloud platform" in the figure. It is observed that OpenStack is the most favorable private cloud platform, occupying 8%, while its competitor OpenNebula and VMware vSphere equally share 1%.

Moreover, the breakdown of the hybrid cloud is shown in Figure 3.7. The sector of "unknown private and public cloud platform combination" denotes that it is not deducible what specific mixture of public and private cloud platform used according

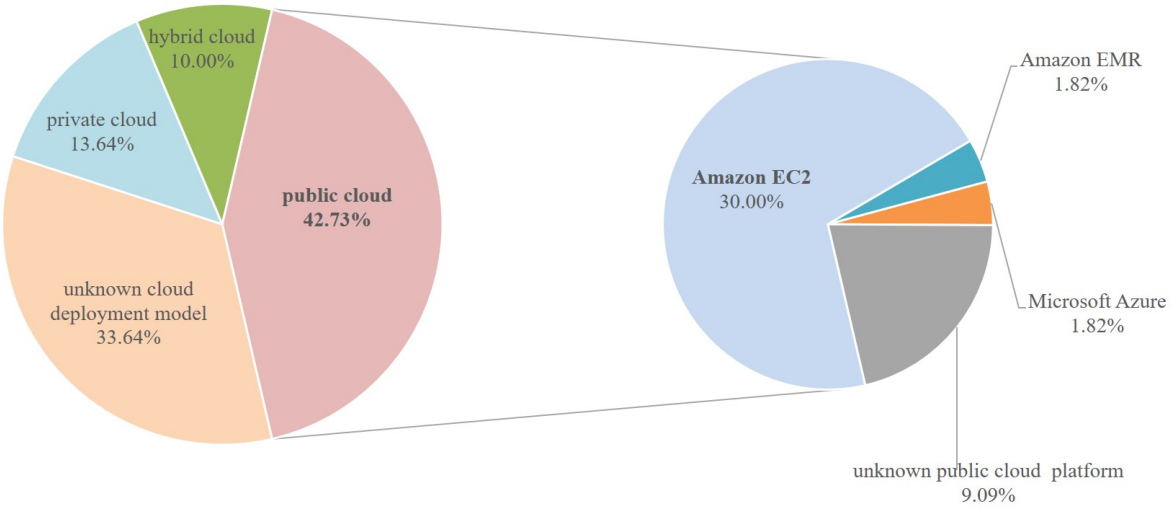


Figure 3.5: Breakdown of public cloud deployment mode

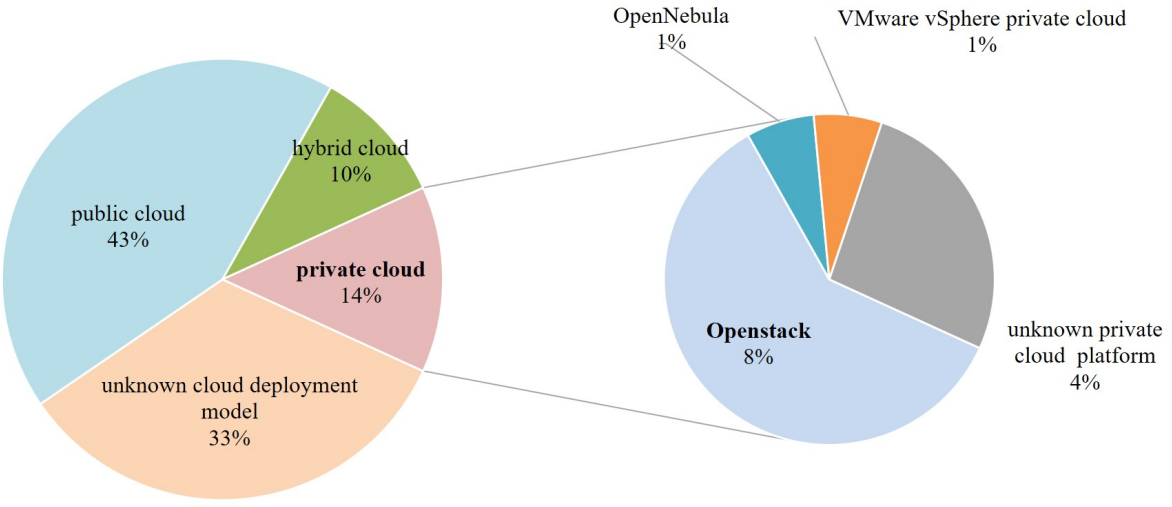


Figure 3.6: Breakdown of private cloud deployment mode

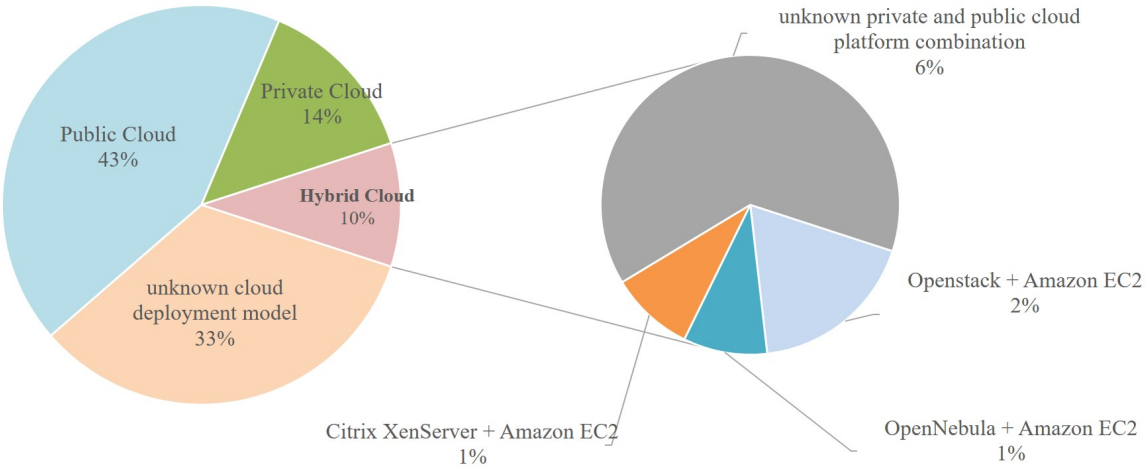


Figure 3.7: Breakdown of hybrid cloud deployment mode

to the reviewed papers. It is interesting to find that OpenStack and Amazon EC2 is the more popular combination despite having a small percentage (2%) than other combination such as OpenNebula plus Amazon EC2, Citrix XenServer plus Amazon EC2 occupying 1% respectively.

3.3.4 Techniques (In response to *SLRQ3*)

Figure 3.8 shows the statistics of the reviewed papers by different SLA management techniques. Based on this figure, it can be deduced that the dominant techniques are optimization, scheduling, simulation, monitoring, machine learning, constraint programming, and scaling. Further, Table 3.5 shows these techniques used to address SLA management for cloud-hosted BDAAs and their breakdown by layer. From this table, it is clear that the most common techniques used in SLA management are Optimized-based, Simulation-based, Scheduling-based, Machine learning-based and Monitoring-based techniques. There are few research works that investigated other techniques such as scaling, fuzzy logic and error-handling, showing that these techniques are uncommon in the landscape of SLA management for cloud-hosted BDAAs.

It is deserving to note that some authors combine more than one technique to address SLA management for cloud-hosted BDAAs. They might use scheduling and

Table 3.5: Techniques used in addressing SLA management for BDAAs in clouds

Technique (papers quantity)	**Service Layer (papers quantity)**	**References**
Optimization-based (11)	BDPaaS (7)	[191, 195, 226, 229, 234, 266, 267]
	CIaaS (4)	[279, 294, 296, 297]
Scheduling-based (35)	BDSaaS (8)	[199, 200, 201, 202, 216, 217, 257, 298]
	BDPaaS (24)	[213, 211, 192, 218, 219, 222, 223, 224, 227, 228, 231, 233, 236, 237, 238, 241, 245, 249, 251, 255, 259, 256, 260, 287]
	CIaaS (3)	[292, 288, 291]
Simulation-based (54)	BDSaaS (3)	[196, 198, 201]
	BDPaaS (36)	[213, 211, 191, 195, 214, 215, 222, 227, 228, 229, 231, 233, 234, 236, 237, 238, 240, 241, 242, 243, 245, 246, 247, 255, 259, 256, 257, 258, 299, 262, 263, 265, 266, 267, 268, 297]
	CIaaS (15)	[292, 277, 279, 281, 282, 283, 284, 288, 289, 291, 293, 294, 295, 296, 297]
Monitoring-based (11)	BDSaaS (5)	[204, 205, 206, 207, 209]
	BDPaaS (4)	[261, 262, 264, 272]
	CIaaS (2)	[277, 280]
Machine learning (10)	BDSaaS (5)	[196, 198, 200, 205, 209]
	BDPaaS (3)	[226, 263, 264]
	CIaaS (2)	[271, 277]
Control-based (3)	BDPaaS (3)	[212, 225, 221]
Constraint Programming (7)	BDSaaS (1)	[203]
	BDPaaS (6)	[235, 240, 242, 246, 247, 248]
Scaling (7)	BDPaaS (5)	[252, 253, 254, 268, 273]
	CIaaS (2)	[280, 282]
Mixed Integer Linear Programming (3)	BDPaaS (2)	[234, 248]
	CIaaS (1)	[294]
Fuzzy Logic (1)	BDSaaS (1)	[196]
Framework-based (4)	BDSaaS (1)	[300]
	BDPaaS (2)	[194, 274]
	CIaaS (1)	[290]
Anomaly Detection (2)	BDPaaS (2)	[277, 285]
Auction (2)	CIaaS (2)	[283, 284]
Model Checking (1)	BDSaaS (1)	[210]
Heuristic-based (1)	BDPaaS (1)	[220]
Policy-based (1)	BDPaaS (1)	[232]
Game Theory (2)	BDPaaS (1)	[250]
	CIaaS (1)	[295]
Error Handling (2)	BDPaaS (2)	[243, 244]

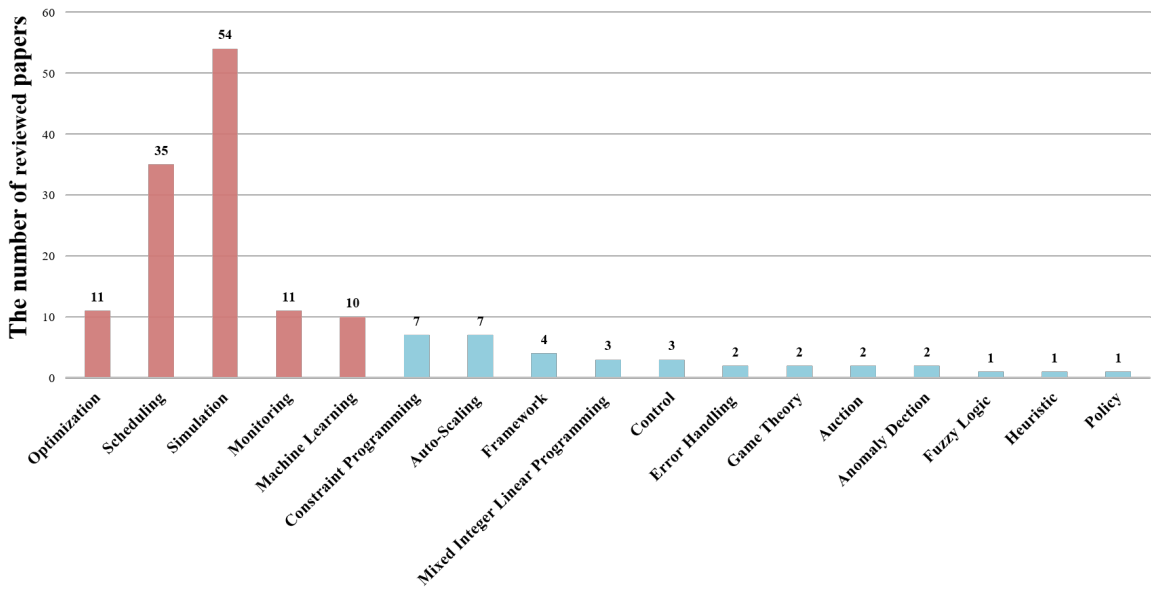


Figure 3.8: Statistics of the reviewed papers by techniques

machine learning-based technique, or fuzzy logic and machine learning-based technique, or monitoring and scaling technique. For example, Rajinder et al. [200] propose a overall architecture regarding SLA-aware scheduling of BDAAs across geo-distributed cloud datacenter. Their proposed scheduling algorithms have two levels including coarse-grained and fine-grained. In this paper, firstly, they employ Naive Bayes algorithm to predict which category a user's BDAA belongs to. They then apply Adaptive K-nearest neighboring-based scheduling algorithm to discover which regional datacenter is appropriate based on locations and requirements of users. Next, they performed the optimal scheduling of big data jobs based on their designed scheduling architecture and typical Amazon scheduling policies in the local server. In this paper, they investigated SLA metrics such as the time of waiting, the utilization of CPU, availability, estimated time to complete and response time. Experiments shows the efficacy of their coarse- and fine-grained scheduling algorithms. In [196], the authors focus on big media healthcare BDAAs in cloud, which must satisfy SLAs for medical users. In this work, they exploited fuzzy logic to orchestrate a local- and global-based cloud federation model that optimizes the selection decision making regarding target cloud data centers. The model considers the trades off between the users' application service quality and

providers' profit when choosing federated data centers. Also, the model acknowledges the dynamic behavior that user requests posses and system environments. Through the precise estimation of resource requirements for processing big data jobs using multiple linear regression algorithms, the accuracy of selection decision is significantly enhanced. In the subsequent subsections, we will give the details of the fundamental properties of some representative techniques and their application in some of the reviewed papers.

3.3.4.1 Optimization-based

Generally speaking, the appropriate utilization of resources makes the tasks of SLA management more favorable for providers. As a result, providers continuously demands optimization-based algorithms that can optimally allocate/reallocate resource to maximize resources utilization and providers' profit. It makes sense that an optimization-based technique is essential in this context.

For cloud-hosted BDAAs, the vast configuration diversity and dependency across different layers makes it difficult for customers to choose appropriate configurations or even decide an applicable background regarding their decisions. Moreover, the extant simple optimization algorithms fail to meet the requirements of most BDAAs that are often featured by different objectives, either because one of the objectives is unsatisfied, or the results appear far from the optimum [301]. Consequently, allocating cloud resources (at CIaaS level) to big data platforms (BDPaaS level) is not any more a conventional single objective problem (e.g., minimizing time, maximizing resource) but involves multiple contradictory objective functions expressed by SLA metrics such as the maximization of classification accuracy using Apache Spark MLlib, the minimization of response time of MapReduce tasks using Apache Hadoop, the minimization of stream processing latency using Apache Storm and the maximization of CPU utilization and so on. Further, the formulated multi-objective optimization problem demands a considerable amount of computation that is increasing exponentially with the problem size in order to find optimum solutions.

Take the energy consumption optimization for BDAAs as an example, the authors [279] propose a multi-objective optimization-based technique that is aware of both energy and SLA requirements when placing and consolidating VMs. Their proposed

technique considers to balance the performance and energy utilization of such system as well as SLA-compliance (e.g., availability and reliability). The results demonstrate that their technique achieves better performance on saving energy, reducing resource consumption and communication cost, minimizing the quantities of VM movements and SLA violations in comparison with the other extant tested approaches.

In terms of optimizing PaaS providers' profit, Dib et al. [191] propose a decentralized optimization-based policy and consider to pay the penalties when SLA violations occur. Their proposed policy achieves optimally exploiting private resources, especially at peak time, before leasing any public cloud resources. The paper applies their proposed optimization-based policy into a realistic batch-based BDAA. Similarly, the authors [284] addressed the challenges of allocating resource while guaranteeing SLAs and maximizing providers' profit. In this paper, the penalty cost incurred by SLA violations is considered in order to increase providers' profit. They take into account SLA metrics such as execution time and deadline of jobs (i.e., urgency) in a combinatorial auction system and propose a new winner determined algorithm (an optimization-based technique). Experiments proves the efficacy of their approach on the reduction of the penalty payment incurred by SLA violation and maximization of providers' profit.

Unlike the above works, paper [296] addresses how to optimize the distribution of big data and allocation of computing resources on mobile cloud platforms. As such, the authors propose a new network architecture and algorithms. They discuss an end-to-end thin-thick client collaboration to efficiently distributing data by transferring large dataset into splits depending on the bandwidth of Internet connection. Also, this paper details the procedure of selecting suitable algorithms that can efficiently enhance the utilization and allocation of resources and improve users experience by meeting expected SLA requirements (e.g., minimized VMs quantity, shortened execution time and budget).

3.3.4.2 Scheduling-based

Scheduling is one of the fundamental techniques in addressing SLA management for cloud-hosted BDAAs. Primarily, this technique is based on the above optimization technique where an Non-determin istic Polynomial-time Hardness (NP-hard) optimiza-

tion problem has been formulated. Unlike the optimization-based technique, scheduling takes a further step of allocation works based on optimal solutions. Depending on different purposes, such allocation works include assigning physical resources (e.g., machines) to virtualized resources (e.g., VMs), or allocating VMs resources to particular batch or stream processing tasks, or designating platform resources to various BDAAs in a smart way while respecting SLA requirements.

Scheduling has been widely used for traditional applications or workflows in cloud computing (CC) environment. However, unlike them, distributed data processing technologies such as MapReduce paradigm are now often utilized by many organizations to deploy their big data analytical applications (BDAAs). Therefore, scheduling algorithms used for traditional applications or workflows in CC environment cannot be applied directly to BDAAs in clouds due to the complexities that data processing frameworks incur and the difference of resource allocation mechanisms that big data brings. As a result, various SLA-based scheduling mechanism and algorithms have been proposed, which primarily aims to optimize resource utilization and provides optimal resource allocation/reallocation solutions for cloud-hosted BDAAs while meeting multiple SLA requirements.

When addressing SLA management for BDAAs in clouds, scheduling technique can be applied at different layers. Hence, it can be classified into three classes, i.e., "scheduling at the BDSaaS layer", "scheduling at the BDPaaS layer" and "scheduling at CIaaS layer".

Scheduling at the BDSaaS layer

Optimally and strategically providing low-level resources to support BDAAs, jobs or workflows while guaranteeing agreed SLAs between providers and customers is the fundamental objective for the tasks of scheduling at BDSaaS layer. Scheduling at this layer has twofold consideration. On the one hand, it should satisfy users' SLA requirements and optimize objectives such as complete time, makespan, user capital expenditure, and application performance from the customers' perspective. On the other hand, it should efficiently schedule big data platform resources to the application layer to maximize profit or reduce the carbon cost or energy consumption by cloud centers from the providers' perspective [302].

Verma et al., [233] consider SLA violation concerning the performance of MapReduce-based BDAAs and propose an automatic framework of resource inference and allocation. According to their proposed framework, they firstly profiled some common performance features such as soft deadline and then estimate the number of resources required for completing jobs to meet the deadline. Their proposed algorithm can efficiently schedule the execution sequence of jobs and determine the resources quantities allocated to these jobs while meeting job deadlines.

The authors [211] addressed the challenge of resource scheduling to optimize profit of providers. To this end, they first proposed a scalable and adaptive policy of admission control. Then, they developed a novel algorithm that can optimally schedule resources according to users' query requests while guaranteeing SLAs on deadlines and budgets, and prompt responses with manageable monetary expense.

Scheduling at the BDPaaS layer

Scheduling at the BDPaaS layer aims at allocating some dependent and independent tasks to VMs in Hadoop clusters. A valid scheduling algorithm can provide the optimal solution of task distribution over various VMs in a cluster depending on the requirements of execution time and availability of resources. An optimal distribution of tasks can minimize the scheduled tasks' average execution time and maximize the utilization of allocated resources. As such, the response time of tasks that are to be processed is minimized and resources consumption is reduced [303].

Wang et al. [219] developed a scheduling algorithm at platform-level for MapReduce-based BDAAs that have two practical SLA constraints (i.e., budget and deadline) over the heterogeneous cloud datacenters. They designed a greedy-based optimization algorithm that can find appropriate VMs from an established pool of different VMs to minimize the job completion time and monetary cost of executing jobs.

Tian and Chen in [230] took into account the entire processing phrases for MapReduce jobs. In this paper, they designed a cost model that formulates the correlation between the input data volume, the MapReduce resources availability, and the Reduce tasks complexity. They performed testing over a limited number of machines to learn model parameters. The proposed cost model can facilitate to make decisions in terms of the optimal amount of resources, the minimization of time under particular financial budget

and the minimization of monetary cost under a specific time deadline. Experiments demonstrates that this cost model achieves decent performance and satisfies SLAs for MapReduce-based BDAAs.

Scheduling at the CIaaS layer

Scheduling at this layer is more relevant with the optimal mapping virtualized resources onto physical resources in a homogeneous or heterogeneous environment and with the optimal use of the underlying cloud resources.

Nita et al. [292] discuss the challenge of transferring big data across various cloud datacenters where the performance of VMs migration and data transfers are affected. They describe an optimized method that can transfer large dataset according to the characteristics of network and take into account the SLA constraints such as minimizing the duration of individual VM migration. They proposed a scheduling policy that consists of two greedy-based algorithms to transfer large dataset. It manages and maintains an SLA-aware network that impacts the performance of cloud. They evaluated the proposed scheduling policy by means of the simulation of SLA constraints at CIaaS layer.

In order to address the issue of file requests' tail latency, the authors [291] employed information flow queue theory to provide an optimal scheduling algorithm for erasure codes-based cloud storage systems (at CIaaS layer). They first designed a model based on k-marriage flow queue. They then built a multi-objective based scheduling strategy to find the optimum depending on SLAs preferences of users. Their solution is featured by the decentralization of the queue form that outperforms the centralization of queue format in terms of the elimination of the overhead of block. Their simulated results showed their approach decently improves tail latency in comparison with the extant approaches of data displacement.

3.3.4.3 Simulation-based

Simulation is also a popular technique used to address SLA management for cloud-hosted BDAAs, which allows providers to evaluate a broad spectrum of components such as workload, processing elements (e.g., MapReduce, Storm), data centers, storage, networking, and SLA constraints.

When providers offer their big data analytics service to customers with awareness of relevant SLAs, they can identify potential issues before introducing them into the operations and focus on service meeting the agreed SLAs. As they define this service, coarse SLAs can be identified and decomposed to identify more targeted SLAs that in turn drive qualification of the feasibility of proposed solutions to meet SLA commitments. This can then be verified through simulation to identify further how other resources are impacted by any shortfall to inform prioritization in addressing any gaps to guarantee SLAs such as maximizing overall resource utilization or reducing idle time. Moreover, the simulation technique is used to predict system performance and further to study SLA impacts in a production environment.

The simulation technique abstracts, models and emulates BDAAs with a wide of components such as workload, or SLA metrics. In cases where the performance of application does not satisfy pre-specified SLAs, algorithms, scheduling, or monitoring are adjusted and further optimized, and corrective and proactive measures are adopted before an issue occurs. Hence, simulation is an essential technique to facilitate SLA management for cloud-hosted BDAAs. Also, the real-world cloud-hosted BDAAs covers a wide array of application domains including healthcare, social media, energy and so on. Each type of these application is characterized by diverse architecture, configuration, implementation and deployment requirements. Experimentation in a real environment such as Amazon EC2 or Microsoft Azure for different BDAAs can be challenging for manifold reasons:

- It is not economical to purchase or lease large-scale datacenter infrastructure that will precisely indicate realistic deployment of BDAA and allow researchers conducting experiments with changing hardware resource and dynamic framework configurations, as well as big data diversities in terms of volume, variety and velocity.

- The experiments are not repeatable, because some variables that are not under the control of the tester may affect experimental results.

- Much manual configuration effort involved especially in a real large testbed

experiment environment that needs dynamic configurations significantly slows down the performance analysis and makes it almost impractical. As a consequence, it is remarkably challenging to reproduce the experiments outcomes.

- The experiments on a real large distributed platform are unrealistic to some degree due to a huge cluster where a considerable of nodes run in different conditions.

In this case, the simulation technique offers significant advantages to SLA management for cloud-hosted BDAAs. For example, researchers can conduct controllable and repeatable experiments by means of simulation technique. Also, it becomes easier to study if SLAs met or breached, and investigate how SLAs is impacted by various resources configuration from different layers in a simulated testbed as compared to a real experiment. Simulation technique makes experiments under various configurations of hardware resources easier and provides insights for practitioners to understand the impact that each design choice is upon to SLA guarantees. They also improve the possibility that researchers can share their simulation environment, which contributes to better hypothesis evaluation and results reproducibility. Finally, researchers can instantiate various processing frameworks of BDAAs and multiple workload scenarios as needed by the virtue of simulation-based technique.

We find that 53 papers among the 109 reviewed papers have applied simulation technique, occupying 49%. In order to investigate what particular simulation tools used, we further examine these 53 papers and find interesting results that (i) some papers generally mention that a simulation-based experiment has been conducted without explicitly stating what particular simulation tool used [195, 240, 242, 243, 263, 285]; (ii) some papers simply state that they developed their simulation tools by Java programming and keep them as proprietary code without giving details [288, 291, 231]; (iii) Other papers give specific description regarding simulation tools used [211, 201, 233, 247, 255, 259, 289]. For the first two cases, we label them as "unknown simulation tools". For the third case, we further find that three types of simulation tools are often used in the reviewed papers. They are discrete event simulator (DES) [304, 305], MRPerf [306] and Yarn Scheduler Load Simulator (SLS) [193]. Figure 3.9 shows the distribution of these three types of simulation tools.

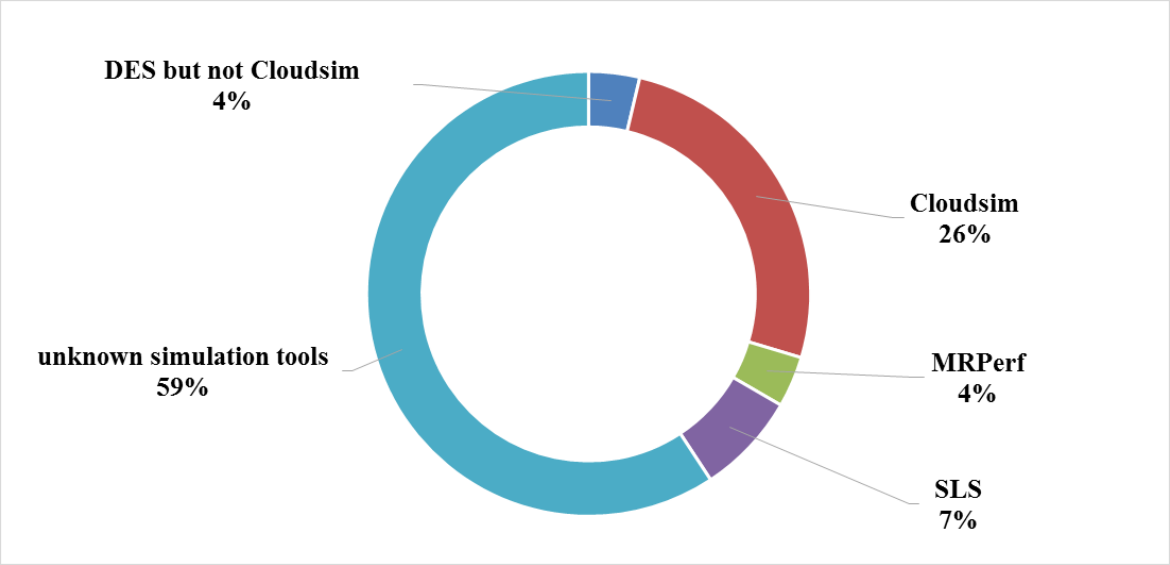


Figure 3.9: Statistics of simulation technique used in the reviewed papers

It is observed that DES is the preferable simulation tool, with a percentage of 30%. Among DES, Cloudsim is the dominant one (26%) and widely used by researchers in their papers. Lots of authors use Cloudsim to implement their algorithms to emulate the CC environment or further implement additional logic to mimic the behavior of the MapReduce model. The second-ranked simulation tool is SLS [201, 227, 255, 256], which can support the simulation of large Yarn clusters and application loads in an individual machine. It is interesting to find that two papers use MRPerf [233, 258], a simulator dedicatedly designed for MapReduce jobs to understand how they perform and study the impact of SLA on various Hadoop configuration settings.

3.3.4.4 Monitoring-based

Monitoring is also an important technique used to manage SLAs for BDAAs. Generally, customers rely on SLAs to deliver the promised quality and level of service. Al through it is readily to notice non-availability or downtime, other types of SLA violation such as performance degradation of VMs and high error rates of APIs are not always easily detected, which can considerably impact the experience of end users. Therefore, monitoring is critical to assure the conformance of SLA and produce fundamental audit trail when SLA violation happens. Moreover, monitoring is essential

for providers to guarantee SLA and offer the satisfactory experience to customers [307].

Monitoring of cloud-hosted BDAAs involves dynamically tracking SLA metrics related to physical resources they share, virtualized resources at CIaaS level (i..e, VM, network and storage), big data processing framework such as Hadoop cluster at BDPaaS level as well as various applications (e.g., smart health, stock recommendation system) running at BDSaaS level. Monitoring is an essential technique to manage SLAs, assisting providers in (i) optimizing the operation of their applications and resources; (ii) capturing performance deviation of application and resources consumed; (iii) monitoring key performance indicators of the applications; (iv) accounting SLA violations regarding specific SLA metrics.

Andreolini et al. [209] take into account the cost minimization regarding computation and communication while assuring peak accuracy in detecting pertinent variations of system behavior guaranteed. They developed an algorithm that can elastically and reliably monitoring big data, which can adapt to update frequencies and sampling intervals. They used real-time series to perform experiments, which shows that their proposed algorithm outperforms extant algorithms in terms of reducing monitoring overheads and maintaining data quality.

The authors [206] develop a method that exploits runtime monitoring to guarantee the applications' performance. They implement a monitoring framework, which collects monitoring data at runtime in a realistic cloud environment. They then design a performance model that uses data mining techniques to extract from authentic monitoring data at runtime. This model sheds lights on how to adjust the strategy of provisioning resources under specified performance-based SLA requirements.

In the context of monitoring stream-based events in a complicated and time-constrained system, the authors [262] design a framework that can real-time monitor and process large-scale log file streams from various sources. They applied the central limit theory to verify soft deadlines in a real-time system and used the probabilistic deadline to ensure SLA satisfied regarding deadline. Flume is used to collect, aggregate, and transfer voluminous stream-based data from multiple sources to a centralized place where Hadoop HDFS is operated. They extended a generic monitoring architecture and illustrated how to calculate the likelihood of SLA violation. This solution is beneficial

for a system of real-time monitoring to determine the deadline for SLAs compliance.

3.3.4.5 Machine Learning-based

Some of reviewed papers use machine learning techniques to study SLA management for cloud-hosted BDAAs from different aspects. Not only is machine learning used to predict the prospective behavior of resources, but also to detect SLA violations. Machine learning-based technique provides machine derived intelligence to the task of SLA-driven optimization and configuration dependencies across multiple layers. It allows continuously learning many complex behaviors and interactions among interrelated objects/entities in BDAAs scenario and taking the guesswork out of many aspects involved in meeting SLAs more efficiently and cost-effectively. For example, collecting large data regarding VMs, storage, and network at CIaaS layer, Hadoop cluster at BDPaaS layer, and applications at BDSaaS layer, then feeding these data to machine learning-based system, finding the hidden patterns and fixing issues before they might violate SLA guarantee. As long as this wealth of data is gathered, processed and analyzed, machine learning-based technique can learn what constitutes normal behaviors, and it is this baseline that gives the system the ability to detect anomalies and find causes automatically. Thus SLA violation can be avoided. Also, it can simulate and predict the impact of making certain changes in resources and their allocations, which can be particularly useful for meeting SLA objectives such as maximizing resource utilization.

Lama et al. [226] developed a Hadoop-based system that can allocate diverse cloud resources and automate multiple Hadoop parameters configuration while guaranteeing SLAs requirements such as minimal monetary cost incurred. It addressed the major issue of providing MapReduce-based BDAAs under different performance deadlines. Their approach consists of a novel framework including two phrases (machine learning-based offline phrase and optimization-based online phrase). The offline phrase clusters various Hadoop jobs by using Support Vector Machine algorithm. The clustering result is then regarded as an input and fed into the subsequent online phrase. The online phrase exploits optimization-based techniques to assign cloud resources and automate the configuration of Hadoop parameters.

The authors [264] address the challenge of optimal resource provisioning for scalable

BDAAs. Firstly, they identified that most of the applications running inside JVMs such as Spark highly demand effective memory resources. Then, they consider applications that is featured by their SLAs (i.e., relative delay) and apply Random Forest algorithm to predict their valid memory requirements. The prediction approach can uncover the hidden behavior of BDAAs' memory consumption and forecast dynamic prospective memory utilization in distributed cloud environments.

3.3.5 SLA Metrics (In response to *SLRQ4*)

In this element, we examine SLA metrics accounted for in the reviewed papers to find out what SLA metrics have been discussed and how often they are discussed. Table 3.6 summarizes SLA metrics and describes their measurements.

Further, Figure 3.10 present the pictorial representation of the frequency of the above SLA metrics that have been discussed in the reviewed papers. It is observed that the most studied SLA metrics are performance, deadline, resource utilization, and cost. This is consistency with our understanding that actors (i.e., Providers, Customers, and End Users) care more about SLA metrics regarding deadline, cost, performance and resource utilization in cloud-hosted BDAAs. The least studied SLA metrics are serviceability, consistency, elasticity, security, capacity, reliability, and scalability. This is because these SLA metrics have limitations regarding measurability [308]. The medium level discussed SLA metrics include profit, budget, energy, availability, fault tolerance, and accuracy. These category of SLA metrics are attracting increasing interest from researchers.

It is also found that the above SLA metrics scatter in the reviewed papers, without an organized and clear categorization. Therefore, it is necessary to examine SLA metrics for cloud-hosted BDAAs through the consideration of building a clear categorization scheme while respecting BDAA characteristics, such that providers and customers will benefit from this categorization scheme when making conventions and engineering SLAs between them.

Table 3.6: SLA metrics and their measurement for BDAAs in clouds

SLA Metrics	**Unit of Measure**
reliability	the number of concurrent failures that are tolerable, MTTF and MTTR and produces the storage system MTTF, number of successful responses in percentage
energy	cost per kWh, power (watts)
capacity	the capacity that the cloud storage system can store. The options range from kilobytes to zettabytes
availability	percentage of service uptime or downtime
serviceability	period of an outage, duration between consecutive service failures, time to switch over from a failure, time to completely recover from a service failure
security	the ability to detect or tolerate malicious attack
consistency	the degree of equality between responses to queries issued by BDAAs
scalability	the ability to horizontally increase the storage or processing capacity or throughput, and the ability to add more resources (e.g., more processors, memory, bandwidth) to each node to increase capacity or throughput vertically
elasticity	the ability to dynamically and rapidly adjust resources to absorb the demand
fault tolerance	the percentage of continuing operating properly when failures (e.g., data node of Hadoop is down, Map or Reduce task fails) occur
accuracy	percentage of accurate prediction or analysis
cost	monetary cost in terms of VM computing per time unit, electricity prices
profit	revenue made per request
budget	upper bound on monetary cost (dollars) to complete data processing tasks
deadline	upper bound on time (hour) to complete data processing tasks
performance	• throughput: MB/sec • latency • data freshness • time pertinent: waiting time/ response time/execution time /job processing time/job completion time
resource utilization	• CPU pertinent: MIPS, number of cores regarding CPU or vCPU, CPU utilization etc,. • memory pertinent: MB/GB, memory utilization etc,. • time pertinent: waiting time/ response time/execution time /job processing time/job completion time • storage pertinent: storage size, I/O throughput etc,. • network pertinent: bandwidth, data transfer time etc,.
request arrival rate	request per second, arrival rate factor (user side) etc.,
quantity pertinent	• the quantity of jobs • the quantity of tasks (i.e., Map or Reduce) • the quantity of working node allocated for batch-based or stream-based processing • the quantity of disks • input data size • the quantity of required parallel threads • the quantity of replicas • the quantity of data blocks • the quantity of VMs

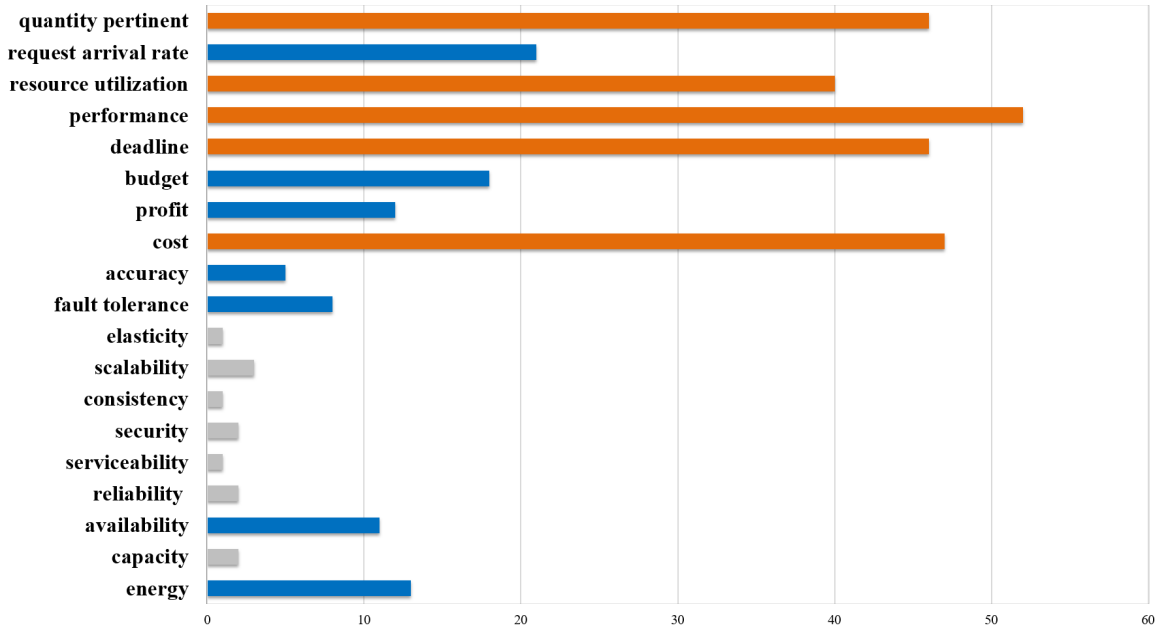


Figure 3.10: Statistics on the frequency of SLA metrics in the reviewed papers

3.3.6 Conceptualization (In response to *SLRQ5*)

3.3.6.1 Conceptual SLA Model

In Cloud Computing (CC) environment, designing conceptual SLA models or frameworks are often discussed. Alhamad et al. [309] proposed a conceptual SLA framework for CC environment. They consider four types of cloud service (i.e., IaaS, PaaS, SaaS, storage as a service). For each different SLA, they present the fundamental parameters that are needed to establish a steady form of negotiation and conversation between customers and providers. Based on the above work, the authors [225] developed a new conceptual SLA model in CC environment called SLA as a Service (SLAaaS). SLAaaS can systematically and transparently integrate service levels and SLAs into cloud. It considers the quality of service levels and SLA as the most superior elements in cloud services.

Also, Labidi et al. [310] proposed a generic and semantic-rich model that is based on ontology theory in their paper. They developed a prototype to validate their proposed model. Through this prototype, the evaluation and triggered guarantee actions of SLAs

can be automatically achieved during their monitoring process.

Moreover, in order to seek an optimal trade-off between revenues and costs while meeting SLA constraints, the authors [311] designed a service-based model that consolidates the major characteristics and SLA objectives of cloud services. Although this model is generic and abstract, it is beneficial to derive a universal and automatic manager with the capability of managing any cloud service, no matter what the layer.

In addition, the authors [312] proposed a formal model to describe SLA contents in CC environment and design autonomic mechanism of predicting SLA violation. Their proposed SLA model is devoted to formalizing a capability to manage SLAs violation detections for cloud services. The proposed approach concerns the representation of information from both the SLAs and cloud logs in a specific format.

All the above-mentioned works are confined to common cloud service without specific consideration of BDAAs. To the best of our understanding, the conceptualization of SLA model dedicated for cloud-hosted BDAAs is nearly empty. This motivates us to propose a new conceptual SLA model for cloud-hosted BDAAs in Chapter 4.

3.4 Summary

In this Chapter, we conducted an in-depth systematic literature review on SLA management for cloud-hosted DBAAs. We propose a novel thematic taxonomy to give an overview of this field. To the best of our knowledge, this is the first systematic literature review specifically on SLA management for cloud-hosted BDAAs. Not only does this literature work provide future researchers an outlook on SLA management for the applications are explored, but it also provides insight into understanding the different research perspectives (Actors, Layers, Techniques, Cloud deployment models, SLA metrics and Conceptualization) of many works in this area.

In the next chapter, we will address the research question defined as *RQ2*.

Chapter 4

A New SLA Model and Categorization Scheme of SLA Metrics for Big Data Analytics Applications in Cloud

In this chapter, we will address the research question defined as *RQ2: How to describe and model service level agreements for big data analytics applications (BDAAs) in cloud in a unified and structured way, and specify common and niche SLA metrics while respecting characteristics of BDAAs?*

According to Section 3.3.6, it is seen that a conceptual SLA model specific for cloud-hosted BDAAs is hardly addressed, which prompt us to fill this gap in this Chapter. We therefore designed a novel cross-layer SLA model for cloud-hosted BDAAs (named CL-SLAMfBDAAs) tailored for this context (see Section 4.1 and 4.2), and proposed a new categorization scheme of SLA metrics for cloud-hosted BDAAs (See Section 4.3 and 4.4).

4.1 Design Principles and Requirements

According to the layered architecture of cloud-hosted BDAAs in Figure 2.2, it is easy to figure out that each layer has two kinds of requirements that are crucial to service composition, which are functional and non functional requirements (FRs and NFRs). These requirements clearly define what the service provider should meet and provide to customers. Figure 4.1 shows the categorization of requirements for layer-based BDAAs in the cloud.

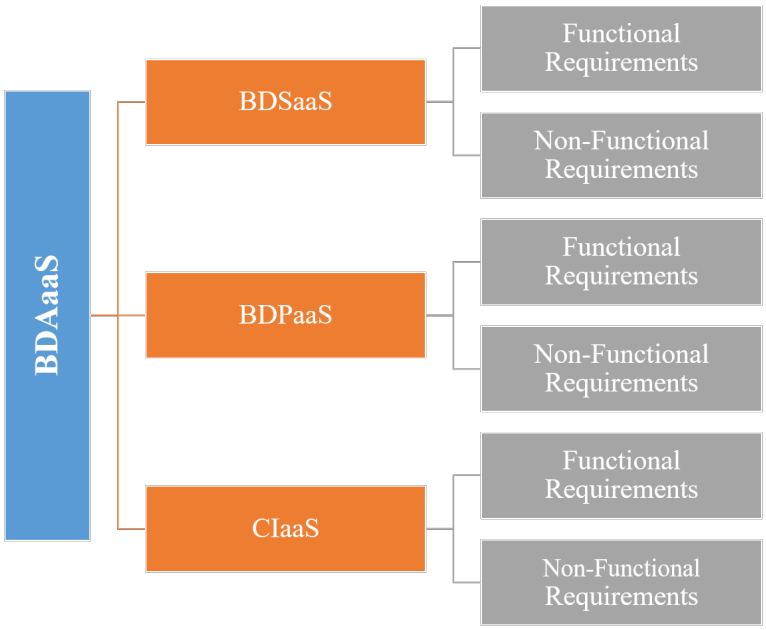


Figure 4.1: Categorization of requirements for layer-based BDAAs in cloud

The focus of FRs is on the functionality of the composed service. For instance, a sentiment analysis service from customer reviews using Amazon Comprehend detects sentiments in the text and extracts information about users' sentiment polarity (Positive, Negative, Neutral or Mixed) [313]. One of FRs in this case is that a sentiment analysis accuracy lower than 80% will never be purchased. A user requests FRs at the top layer (i.e., BDSaaS) and these requirements will be drilled down to the bottom layer (i.e., CIaaS), which provides concrete, scalable and on-demand cloud resources. In other words, upper layer demands resources from a lower layer while a lower layer supplies resources to an upper layer. Thereby, each service in a layer is featured by unified interfaces by which cloud-hosted BDAAs invoke possible functions. For example, Amazon EC2 acts as basic cloud infrastructure and provides a functional interface that supplies its client (i.e., BDPaaS) with computing instances, to install and run software on these instances. By demanding scalable cloud resources from Amazon EC2, Amazon EMR located at BDPaaS layer can supply its client (i.e., BDSaaS) a fully managed Hadoop cluster in minutes, and then Amazon Comprehend uses advanced techniques such as machine learning and natural language processing to predict sentiments as

much accurate as it can. As a result, FRs specified by the users could be met.

On the other hand, NFRs are concerned with SLA metrics, An instance of a NFR for the above Amazon Comprehend sentiment analysis service is that the service response time to a user should be no more than 5 seconds. NFRs are encapsulated and incorporated into SLAs, where multiple metrics are considered such as maximum data transfer ratio, maximum availability, and minimum network latency. NFRs are important to big data analytics service composition and are often formally expressed in SLAs as part of contracts agreed between providers and customers. Even though FRs are met, unsatisfied NFRs such as slow or unreliable service may still not be adopted for BDAaaS.

In addition to functional and non-functional requirements, the dependency relationships between SLAs across different layers of BDAAs is another critical aspect in designing SLA model for the applications. A sole layer is impossible or struggles to provide either FRs or NFRs, thus it is bound to compromise service quality for customers. Having all layers work jointly, the agreed service quality can be guaranteed in the end. Accordingly, we need a novel SLA model for cloud-hosted BDAAs that should meet the following essential design principles:

- Allowing the definition of both functional and non-functional interfaces that expose SLAs by layer for big data analytics service.
- Representing a seamless integration between SLAs and BDAaaS across layers.
- Considering SLAs for cloud-hosted BDAAs in a unified and structured way.
- Reflecting strong dependency relationships between those SLAs.
- Possessing an universal applicability regardless of BDAAs.

4.2 Proposed Cross-layer SLA Model for Cloud-hosted BDAAs

Keeping the aforementioned design principles in mind, we propose a novel cross-layer SLA model for cloud-hosted BDAAs (named CL-SLAMfBDAAs) shown in Figure 4.2.

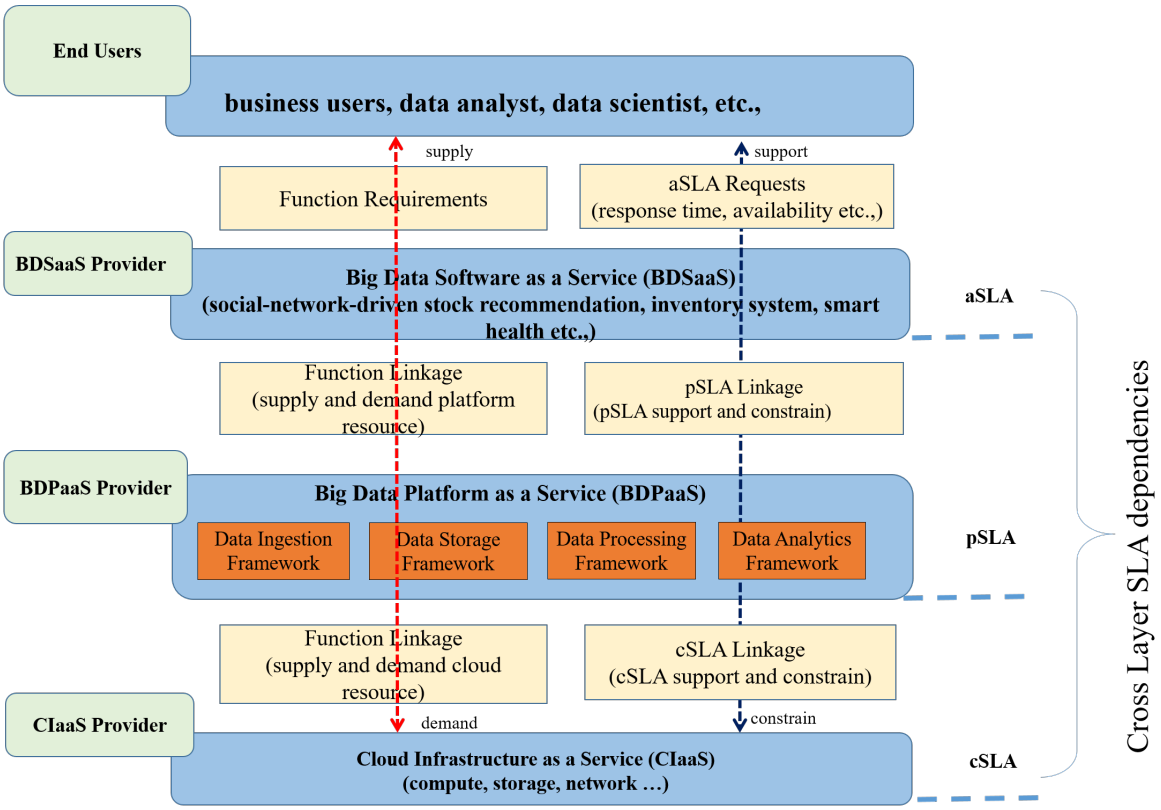


Figure 4.2: Cross-layer SLA model for cloud-hosted BDAAs

As shown in this figure, there are four interacting actors located at different layers. They are end users (e.g., Business Users, Subject Matter Expertise, Data Scientist or Data Analyst), BDSaaS provider (e.g., Salesforce or BrandsEye), BDPaaS provider (e.g., Google Cloud Dataflow, Amazon EMR or Microsoft Azure HDInsight), and CIaaS provider (e.g., Google Compute Engine, Amazon EC2 or HP Cloud). These actors are involved in a set of activities, for instance, negotiating, clarifying and specifying FRs and NFRs, and formulating NFRs into agreed SLAs in each layer. Moreover, it is observed from this figure that SLAs are dedicatedly divided into three categories including application-level SLAs (aSLA), platform-level SLAs (pSLA) and cloud infrastructure-level SLAs (cSLA).

Furthermore, there are strong bonds between the upper layer and the corresponding lower layer of SLAs. From the perspective of FRs, the actor at each layer has a bidirectional relationship with its neighbor's layers either by demanding or supplying

behavior. In other words, the upper-layer actor demands or requests resources from the lower-layer actor while lower-layer actor supplies the resources requested to upper-layer actor. From another perspective, a two-way relationship is existed, where SLAs at each layer either constrain or support SLAs in its adjacent layers. Concretely, an end user requests FRs along with NFRs through the interface with BDSaaS provider. The NFRs will be negotiated and defined into aSLAs between them. Then, an aSLA will be interpreted and formulated into a set of pSLAs. After that, a pSLA will be transformed into a set of cSLAs. From topmost to bottom, the upper-layer SLA sets the constraints into its lower-layer SLA and decides how well the lower-layer SLA must work to meet service-level objectives in the end. In turn, the low-layer SLA works hard to support its upper-layer SLA. For instance, if the availability in aSLA is 99% (a 99% availability at this layer means that users will be able to access the application at least 99% of the time), then, the availability in pSLA must be hover somewhere between 99% and 99.99%, and the availability in cSLA must be higher than the availability in pSLA. In the absence of meeting this, it might fail to guarantee SLA constraints.

In terms of SLA metrics, cSLA metrics such as VMs quantity, CPU and memory resources utilization, or the availability of VMs affects the pSLA metrics such as the quantity of map and reduce nodes in Hadoop platform (Data Processing Framework at BDPaaS layer) or the other pSLA metrics such as the quantity of data nodes, transfer rate, and replication factors of NoSQL database service (Data Storage Framework at BDPaaS layer). Certainly, this at last affects aSLA metrics such as capital cost, availability and reliability of the applications. This exemplifies the strong cross-layer SLA dependency relationship. To guarantee the final SLA to customers, BDSaaS provider should guarantee aSLAs by interweaving pSLAs and cSLAs.

Finally, our proposed CL-SLAMfBDAAs model presents a unified and structured scheme to describe and interpret SLAs for cloud-hosted BDAAs. In this novel model, SLAs are exposed and linked in a vertical motion, which is orthogonal to the layers and may apply to any of them. Based on this model, users know what different types of SLAs with various attributes exist and how they work collaboratively across layers to ensure the delivery of SLA guarantee. This model meets all the aforementioned design principles and requirements. It is further elaborated to propose a new categorization

scheme of SLA metrics for cloud-hosted BDAAs.

4.3 Multi-dimensional Categorization Scheme of SLA Metrics for Cloud-hosted BDAAs

The typical SLA metrics at CIaaS layer that customers expect are the number of VMs, memory size, CPU usage, hard disk utilization, memory usage, additional network parameters and so on. While at BDPaaS layer, the example of SLA metrics include throughput, response time, and availability. For instance, in the case of a Hadoop cluster (at BDPaaS layer), we have metrics such as execution time, job turnaround and makespan in terms of MapReduce tasks [314]. At BDSaaS level, a particular SLA metric is highly determined by the genre of BDAA. For example, the rate and quality of data transfer are important for any video streaming-oriented BDAA, while latency of processing and network generally interests a batch-based BDAA. It is worth to note that SLAs at each layer might have an endless variety of metrics depending on different components and the nature of applications.

Figure 4.3 shows our proposed extensible and multi-dimensional categorization scheme of SLA metrics for cloud-hosted BDAAs. This schema not only defines the most commonly used metrics for each type of SLA (i.e. aSLA, pSLA and cSLA), but also presents niche SLA metrics to be consistent with different components at each layer.

4.3.1 aSLA Metrics

There are lots of different types of BDAAs across a wide range of industries. For instance, topic detection and tracking applications, large-scale log analysis applications and business intelligence. Due to their wide variations, listing all SLA metrics at this level is impracticable. Hence, we select some typical cloud-hosted BDAAs and present common aSLA metrics for them as shown in Table 4.1. To embody the unique features of these BDAAs, we provide niche aSLA metrics for them in Table 4.2.

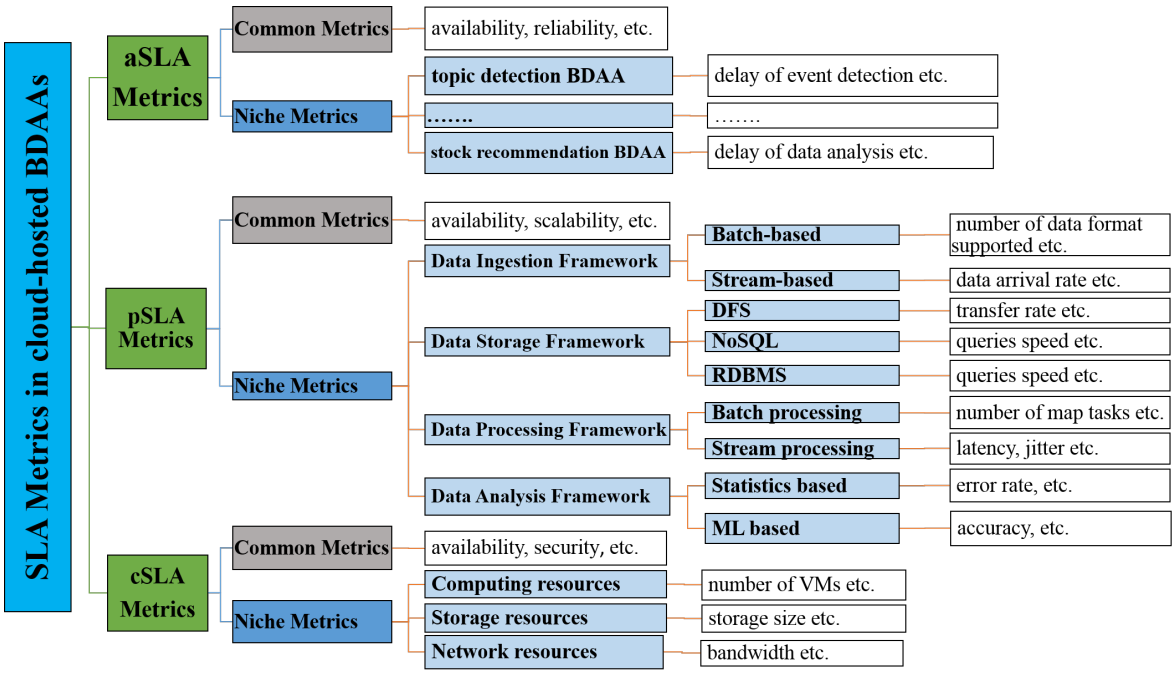


Figure 4.3: Categorization scheme of SLA metrics for BDAAs in clouds

Table 4.1: Common aSLA metrics

aSLA Metrics	**Description**
availability	The uptime of BDAA for end users in a specific time frame
financial cost	The total financial cost of using BDAA
respone time	Time to complete and receive the analysis result
usability	The degree to be easily used by end users through built-in interfaces
deadline	The total time of executing a BDAA and returning final results to its end user
reliability	The ability to maintain operational status in the majority of cases
integration	The degree of simplicity for integrating with applications and services require data from BDAA
capacity	The capacity the BDAA can provision
scalability	The ability to scale when expanding large volume of data or vast number of users
customizability	The flexibility to use with diverse kinds of users
pay-per-use billing	The ability to charge based on the usage of resources or duration
security	The degree to be exempt from malicious attack incurred by the network, software, tools, process or human, which results in significant damage or loss
energy efficiency	The degree of overall energy consumption on a per unit level (e.g., per capita, per customer, per hour)
the ratio of the admitted workloads	The proportion between the permitted workloads quantity and the submitted workloads quantity by end users

Table 4.2: Niche aSLA Metrics by different types of BDAAs

BDAA	aSLA Metrics	Description
Topic detection and tracking application [315, 316]	• event detection delay • input throughput • output throughput	• the delay of detecting events such as earthquake, football matches • the number of input events that are processed during a period • the number of derived events that are produced during a period
Big data-based traffic congestion detection system [317]	• alert sending delay	• the delay of sending alerts of existing traffics
Large-scale ingestion of analytics events and logs application [318]	• log integrity • alert sending speed	• the percentage of logs that can be seen • the number of alerts sent per second
Business intelligence on big data [80]	• the delay of decision making process	• the delay of a decision that is made based on business intelligence
Social network driven stock recommendation system [179, 180]	• data analysis delay	• the delay of completing stock analysis based on social network data
SLA based healthcare big data analysis and computing in cloud network [198]	• disease prediction accuracy	• the degree to accurately forecast patients' prospective disease condition
Google smart inventory management system [82]	• inventory accuracy	• the degree to grasp accurate information regarding inventory and product at any time

4.3.2 pSLA Metrics

There are four main components/frameworks at BDPaaS layer (i.e., data ingestion, storage, processing and analysis). Selection of the specific software or tool as an instantiation of the aforementioned different frameworks is based on many aspects such as flexibility, control and ease of use. Considering the differential nature and role-playing, each framework at BDPaaS layer has different pSLA metrics. Table 4.3 lists some common pSLA metrics, while some niche pSLA metrics for each framework are given in Table 4.4.

4.3.3 cSLA Metrics

Companies like Microsoft, Google and Amazon offer infrastructure as a service. With this diverse range of cloud infrastructures, most customers are perplexed to choose which SLA metrics should be defined and specified as the hardware section of cSLAs. To clear away this confusion, we give the most common and niche SLA metrics that interest customers when using cloud resources in Table 4.5 and Table 4.6 respectively.

Table 4.3: Common pSLA metrics

pSLA Metrics	**Description**
availability	The uptime of each framework in BDPaaS in a specific time
integration	The abilities to integrate with other frameworks and platforms
capacity	The capacity the BD platform can provision
scalability	The abilities to expand platform-level resources as requests or workloads increase
pay-per-user billing	The ability of the charging based on which framework or time of utilization
energy efficiency	The degree of energy consumption on a per unit level (e.g., per capita, per customer, per hour) for each framework
security	The degree to be free from malicious attack incurred by software or tools in each framework at BDPaaS layer, which brings damage or loss
fault tolerance	The ability to maintain an appropriate operational status even in the case of failures within its components

To the best of our knowledge, this is the first attempt to systematically categorize SLA metrics for cloud-hosted BDAAs. This new categorization scheme cannot only demonstrate SLA metrics for cloud-hosted BDAAs in a clearly structured format but also respect the characteristics of cloud-hosted BDAAs. The categorization scheme facilitates design decisions, analysis of existing SLAs, making conventions and engineering SLAs between providers and customers, and helps to identify responsibilities during the management of SLAs.

To better understand SLAs across different layers of BDAA, we present a SLAs template using a real cloud-hosted BDAA in the following section.

4.4 Detailed Example of SLAs Template for a Real Cloud-hosted BDAA

To aid understanding of the SLAs across different layers of BDAA, we take a real cloud-hosted BDAA as an illustrative example to present SLAs template. The selected BDAA example is a smart inventory system for retail offered by Google as one of their referred BDAA solutions and use cases [82].

Table 4.4: Niche pSLA metrics by the different framework at BDPaaS layer

Framework	**Sub Category**	**pSLA Metrics**
Data Ingestion	Batch-based	• data size • the number of chunks • chunk size • throughput
	Stream-based	• data size • data arrival rate • latency • the number of data format supported
Data Storage	Distributed File Systems (e.g., HDFS)	• transfer rate (read, write) • latency (read, write, update) • the number of data nodes • replication number • the overall size of input data • the size of split data • network throughput
	NoSQL	• transfer rate (read, write) • latency (read, write, update) • the quantity of data nodes • replication factors • queries speed • transaction response time • data freshness
	RDBMS	• queries speed • query throughout • the number of connections • buffer pool usage • transfer rate (read, write) • latency (read, write, update) • replication factors • batch requests/sec • disk read I/O per sec • disk write I/O per sec
Data Processing	Batch-based	• the number of map tasks • the number of reduce tasks • the number of unhealthy nodes • the number of active nodes • the number of instances • upper bound on the time finishing the data processing task • block size • job turnaround • maximum allowed completion time
	Stream-based	• response time to streaming data • stream processing latency • peak system resource usage • system start-up time • Jitter (the variance of processing times)
Data Analysis	Statistics-based	• how good is the statistical method (error rate, sensitivity, validity)
	Machine learning-based	• how good is the machine learning model (precision, recall, accuracy, sensitivity, specificity) • model training time • model training speed • the size of the machine learning model • average response speed for individual prediction requests • number of algorithms supported for data analysis

Table 4.5: Common cSLA metrics

cSLA Metrics	**Description**
availability	the uptime of cloud infrastructure in specific time
capacity	The capacity that the cloud infrastructure can provision
scalability	The ability to expand infrastructure-level resources (e.g., VMs) requested from BDPaaS level
pay as you go billing	The ability to charge based on time of utilization of VMs or storages
energy efficiency	The degree of energy consumption for data centers
security	the degree to be exempt from malicious attack incurred by cloud infrastructure, which causes damage or loss

Table 4.6: Niche cSLA metrics by different components at CIaaS layer

Component	**cSLA Metrics**
Computing resources	• response time • CPU utilization • memory utilization • system load • scale up time • number of VMs • number of cores per CPU • memory size • duration of individual VM migration
Storage resources	• number of units of data storage • storage size • privacy • backup • hard disk utilization • I/O speed (bytes per second) • maximum downtime
Network resources	• network throughout • network bandwidth • network latency • accessibility to the Internet across the firewall

According to Google, a smart inventory system for retail are built to maintain an accurate and update information as anything changes in the inventory. For example, as soon as a retail sold an item or even when moved one part of store to another, inventory data should be refreshed in backend automatically. This brings lots of challenges for retails who increasingly struggle to work in scenarios such as online and offline mode as well as multiple channels. Hence, smart inventory management system comes into the picture to address these challenges, such that, retailers can efficiently work with buyers and assist customers locate products faster. Also, retailers can significantly benefit from operational efficiency by precisely perceiving the location and availability of product

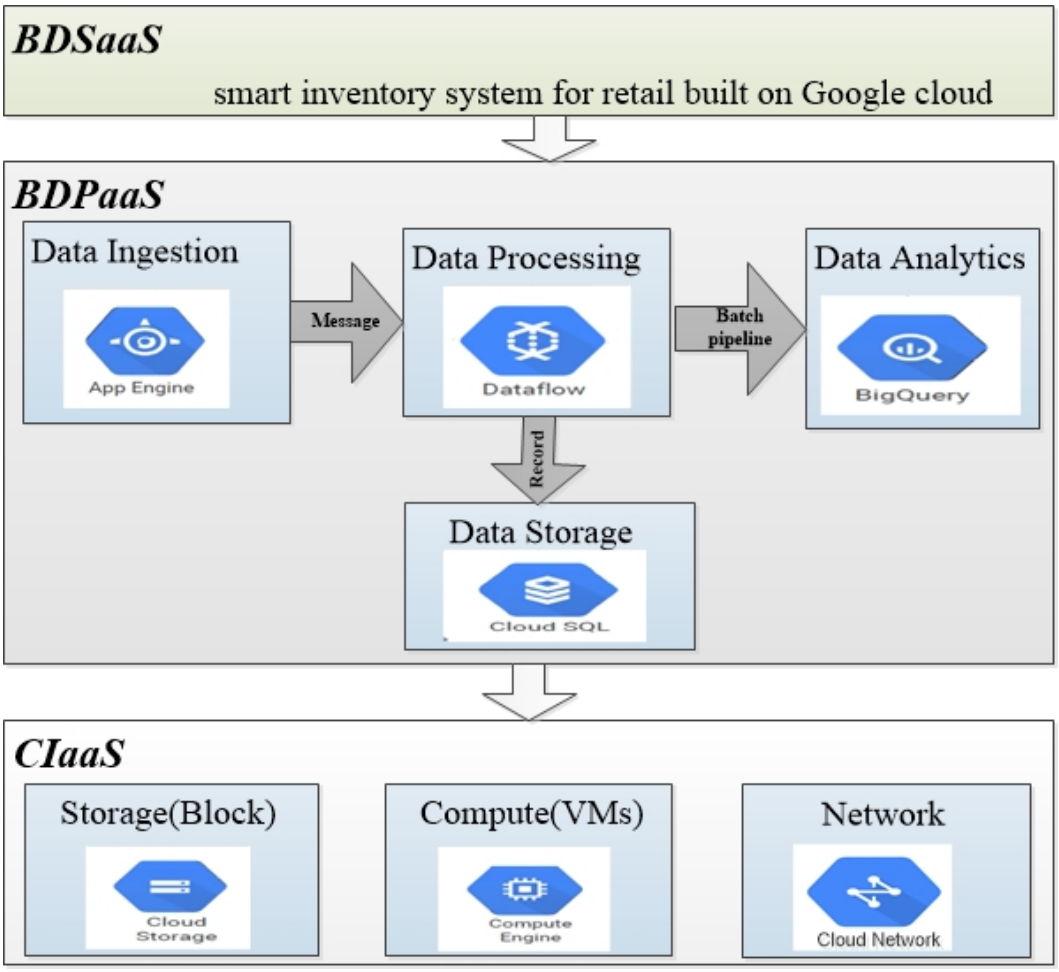


Figure 4.4: Architecture of Google's inventory system for retail

at any time. Moreover, the inventory accuracy and movement offers useful insights for retailers to fine-tune and better promote their products in market campaigns. To this end, retailers need to leverage the power of cloud-based BDAAs that offer scalable infrastructure, reduced administrative complicatedness, and the state-of-art techniques in terms of data ingestion, storage, process and analysis.

Figure 4.4 presents the architect of Google's smart inventory system for retail. The components at each layer, and the interactions and SLA requirements of each component are described as follows:

CIaaS layer

- Compute Engine: It delivers multiple different types of VMs. Users can access Google's VMs using predefined sizes or creating customized machine types to suit particular business needs. These VMs serve the Cloud Dataflow at BDPaaS layer. This component must satisfy SLA requirements. For example, it must ensure availability to the customer at minimum 99.99% [319]

- Cloud Storage: Users can access data stored on Google cloud platform infrastructure through an online RESTful-based file storage web interface. This component must satisfy SLA requirements as well. For instance, it must guarantee at least 99.95% availability to customers [320]. Also, it must guarantee the security and confidentiality of an application and customer data

- Cloud Network: Cloud virtual private network (VPN) offers a secure VPN tunnel to connect customers' own network and Google's global network. One of SLA requirements that Google VPN must meet is to assure minimum 99.9% network availability to customers [321]

BDPaaS layer

- Data Ingestion: This regularly collects inventory data from multiple stores and proceeds it to subsequent services. A good example of data ingestion is Google App Engine that offers the automatic and real-time scaling abilities according to the future traffic patterns. SLA requirements are mandatory for this component. It must provide 99.95% above availability [322]. Moreover, it needs to meet the requirements regarding elastic scaling and minimum response time latency.

- Data Processing: This service includes two popular processing paradigms (i.e., batch or stream) for a smart inventory system. A good example of data processing is Google's Cloud Dataflow that offers the abilities to transform and enrich data both in batch-based and stream-based workloads, and distribute these processing workloads across many VMs instances. In terms of SLA requirements, it must efficiently process batch or stream jobs with low-latency response time. The monthly uptime percentage (availability) defined by Google is 99.5% [43].

- Data Storage: This service is dedicated to record and maintain accurate inventory data at any time. Once new inventory events either through purchase or shipping happen, inventory database is then updated in an automatic and real-time form. Google Cloud SQL is a well-suited choice for data storage, which is deployed in MySQL and provides a native support regarding backup, replication and recovery. Still, SLA requirements are mandatory for this component. For example, it must support high-speed queries for counts from the inventory. In the documentation of Google cloud SQL SLA, the availability is set to be greater than 99.95% [323].

- Data Analytics: This collects incoming inventory streams and load them into BigQuery where specific analysis is performed and actionable insights are generated. BigQuery is a state-of-art Google's data warehouse solution. It can execute queries over TB amount of data within seconds. Without exception, this service must satisfy SLA requirements as well. For example, it must guarantee the availability that is greater than 99.9% [40]. Also, it must guarantee queries speed across TB volume data in seconds, and the high accuracy of the queries result regarding the inventory at all times.

BDSaaS layer. Google's inventory system is designed to provide a high accurate, visible, and analytical platform for inventory movements throughout the supply chain at any time. This service must satisfy SLA requirements such as availability, usability, scalability, integration, response time, financial cost, accuracy and query speed. Customers can then choose service level objectives (SLOs) they want to apply for aSLA (application-level SLA). For instance, response time is managed lower than a designated threshold, financial cost is efficiently controlled without exceeding a given bar, agreed availability level is greater than 95%, and specification of penalties applied when SLA violations incur.

For this application, its SLAs consist of aSLA, pSLA and cSLA. Figure 4.5 presents its cross-layer structure of SLAs metrics. It is worth noting that this cross-layer structure of SLA metrics still applies to other BDAAs although we take the Google smart inventory management system as an instance. Table 4.7 elaborately describes an illustrated example of this application's aSLA, pSLA and cSLA.

Table 4.7: Example of aSLA, pSLA, and cSLA for Google's inventory system

aSLA- SLA between customers and Google		
	SLO	90% of requests handled by Google smart inventory management BDAA will be processed in < 500ms
	Guarantee	Google smart inventory management BDAA must ensure availability at least 99%
	Violation	If the response time of Google's inventory management as agreed is not met during any period, all requests for that period will be free of charge. If availability is < 99%, then 10% of service credit will be applied
pSLA-SLA between BDSaaS and BDPaaS		
Date Ingestion	SLO	For a maximal capital cost of USD 0.056/vCPU per hour (batch type) or USD 0.069/vCPU per hour (streaming type) to the Data Ingestion framework in BDPaaS (e.g., Google App Engine), the maximum response time must be 1 second per ingestion
	Guarantee	The data ingestion service (i.e., Google App Engine) will be available of at minimum 99.95%
	Violation	If availability is < 99.95%, then 10% of service credit will be enforced
Date Storage	SLO	Requests to the Data Storage framework in BDPaaS (e.g., Google CloudSQL), the response time has to be lower than 1 second
	Guarantee	The data storage service (i.e., Google Cloud SQL)will be available of at least 99.95%
	Violation	If availability is < 99.95%, then 10% of service credit will be practised
Date Processing	SLO	Regarding a maximal financial expense of USD 0.056/vCPU per hour (batch type) or USD 0.069/vCPU per hour (streaming type) to the Data Processing framework in BDPaaS (e.g., Google Dataflow), response time has to be smaller than 1 second per dataflow job
	Guarantee	The data processing service (i.e., Google Dataflow) will be available of at least 99.5%
	Violation	If more than 2% of queries to the Data Processing framework(i.e., Google Dataflow) in BDPaaS violate SLOs, the customer will receive the financial credits by 0.02/violated dataflow jobs If availability is < 99.5%, then 10% service credit will be charged
Date Analysis	SLO	Requests to the Data Analysis framework in BDPaaS (e.g., Google BigQuery), response time has to be no more than 1 second
	Guarantee	The data analysis service (i.e., Google BigQuery) will be available of 99.9% at least
	Violation	If availability is < 99.5%, then 10% of service credit will be applied
cSLA-SLA between BDPaaS and CIaaS		
Compute	SLO	For a maximal capital cost of USD 0.0100/hour per VM instance of the cloud infrastructure computing service (i.e., Google Compute Engine), at least 10 VMs, each configured with one vCPU and 3.75GB memory must be provided
	Guarantee	Availability to customer of at least 99.99% should be provided
	Violation	If more than 1% of the time VMs at CIaaS layer has breached SLOs, USD 0.3/hour per the violated VM instance is penalized
Storage	SLO	For a maximal capital cost of USD 0.026 GB/month of the Cloud infrastructure storage service (i.e., Google Cloud Storage), at least 1TB multi-regional storage must be provided
	Guarantee	The data storage service (i.e., Google Cloud SQL)will be available of 99.95% at least
	Violation	10% of service credit will be imposed when availability is lower than 99.95%
Network	SLO	For a maximum financial cost of USD 0.050 per tunnel per hour of the cloud infrastructure network service (i.e., Google Cloud Network), at least 100 tunnels and 100GB bandwidth must be provided
	Guarantee	The cloud network service (i.e., Google Cloud network) will be available of 99.9% at minimum
	Violation	If availability is < 99.9%, then 10% of service credit will be enforced

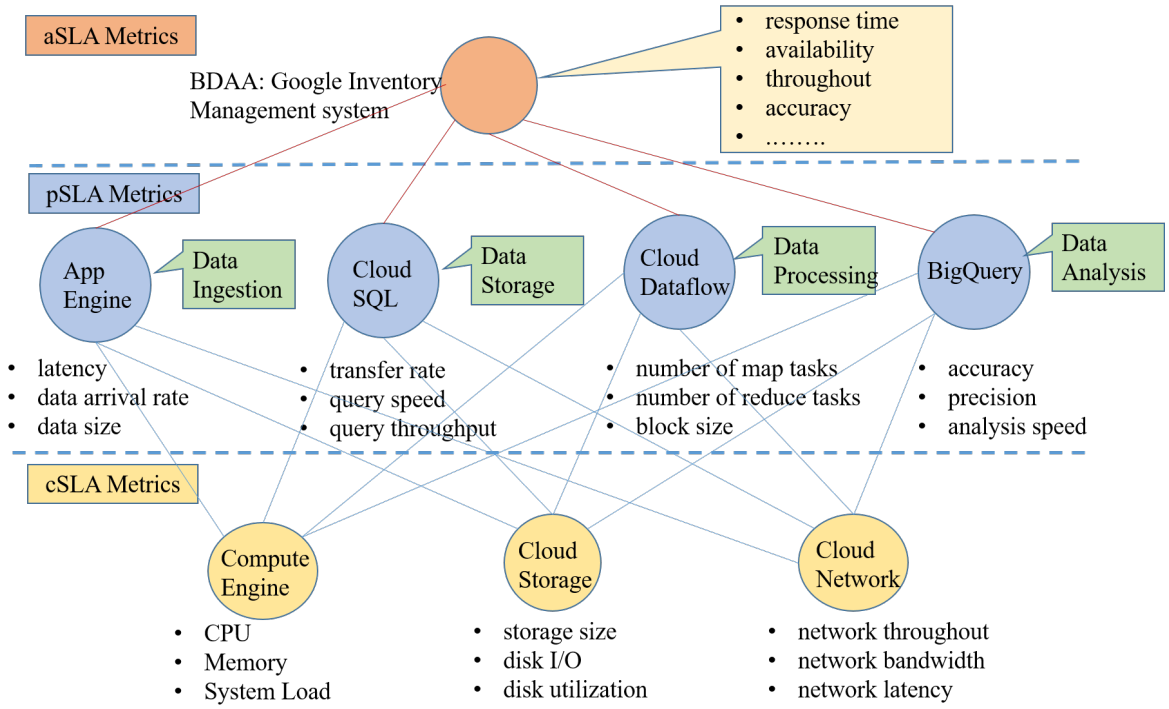


Figure 4.5: Cross-layer SLA metrics for Google's inventory system

4.5 Summary

The contributions of this Chapter are two-fold. Firstly, we propose a new cross-layer SLA model for BDAAs in cloud. This model considers SLAs for cloud-hosted BDAAs from a layer-based perspective, which can not only demonstrate three types of SLA (aSLA, pSLA, and cSLA) in a structured format and a universal way to look at SLAs for the applications but also indicate their strong dependencies relationship. Secondly, we propose a new multi-dimensional categorization scheme regarding SLA metrics dedicated to cloud-hosted BDAAs, while respecting characteristics of BDAAs. This categorization scheme consists of common metrics and niche metrics for each layer. It provides a guideline for providers and customers to understand SLA conventions between them and make design decision in particular concerning the definition and specification of SLAs. In the next chapter, we will address the research question defined as *RQ3*.

Chapter 5

Modeling and Simulation of Big Data Analytics Applications in Cloud

In this chapter, we will address the research question defined as *RQ3: How to model and simulate cloud-hosted big data analytics application (BDAAs) across layers to facilitate service level agreements (SLAs) management?*

To this end, we designed and developed a new discrete event-based simulation tool called IoTSim with the abilities to model and simulate BDAAs in cloud.

5.1 Positioning versus Existing Simulation Techniques

Simulation technique offers many "what if?" scenarios to be evaluated, which allows decision-makers to test and better understand alternative ways in which a new algorithmic approach may be best met. Two main types of simulation techniques have been widely used to study the behaviors of systems. They are deterministic simulation and stochastic simulation. [324]

- Deterministric simulation: refers to simulation models that contain no random variables. Deterministic simulation has a known set of inputs which will result in an unique set of outputs. A typical deterministic simulation is discrete event-based simulation (DES) that models the behaviour and performance of a complex system as an ordered sequence of well-defined "events" [325]. Within the context of DES, an event is defined as an incident which causes the system to change its state only at a discrete set of points in time [326]. For example, a series of events

will be created whenever a MapReduce job requests resources from datacenters, resources are allocated, runtime environment is initializated, the job is executed, and terminated. A succession of these events provide an effective dynamic model of the system being simulated. DES is often used to model deterministic systems and provides a reasonably accurate approximation of a system's behaviour as it does in reality [327].

- Stochastic simulation: has one or more random variables as inputs. Random inputs lead to random outputs. Since outputs are random, they can be considered only as estimates of the true characteristics of a model. In a stochastic simulation, the output measures must be treated as statistical estimates of the true characteristics of the system. Monte Carlo simulation is commonly featured as stochastic simulation. It is related to DES and usually make extensive use of recurrent random sampling in order to obtain statistical results and understand the impact of uncertainty in forecasting models. Monte Carlo simulation is often used to effectively model systems in which probability and non-determinism plays a major role [328].

Since cloud-hosted BDAAs are considered as deterministic systems rather than stochastic systems (e.g., the distributed processing model such as MapReduce in a cloud-hosted BDAA offers deterministic behaviours that are featured a series of well-defined steps such as Map phrase or Reduce phrase). Hence it is not hard to understand that DES is a more appropriate simulation technique used to solve SLA management problem for cloud-hosted BDAAs in comparison with the other two simulation techniques. This is also evidenced by the the SLR that we conducted in Chapter 3 where it is seen that DES is the most widely applied simulation technique by current researchers in this field. Therefore, in this thesis, we determine to focus on DES to further investigate and understand the whole picture and limitations of existing simulation techniques in terms of DES.

Some representative DES simulators in cloud computing (CC) environment are widely discussed. They are GreenCloud [329], iCanCloud [330], FlexCloud [331], CloudSim

[332] and its variants (CloudAnalyst [333], NetworkCloudsim [334], EMUSIM [335] and MDCSim [336]):

- GreenCloud: A packet-level simulator that is capable of modeling behaviors of network links, switches, gateways, and other hardware resources (CPU and storage) in cloud datacenters. The goal of this simulator is to simplify performance tests of energy-aware scheduling algorithms in cloud environments. GreenCloud is a packet-level simulator hence it requires extra memory and processing power to create and transmit packets across simulation entities.

- iCanCloud: A simulation platform which is oriented towards the simulation of a wide range of CC systems and their underlying architecture. It can model and simulate large environments (thousands of nodes) and distributed applications with a customizable level of detail.

- FlexCloud: A flexible and scalable simulator that enables users to simulate the process of initializing cloud datacenters, allocating VMs and providing performance evaluation for various scheduling algorithms. FlexCloud can be run on a single computer with JVM to simulate large-scale cloud environments with a focus on infrastructure, adopts agile design patterns to assure the flexibility and extensibility, models VMs migrations which are a lack in the existing tools, provides user-friendly interfaces for customized configurations and replaying.

- Cloudsim: A generalized and scalable simulation tool written in Java and was initially built on top of SimJava and GridSim [337]. SimJava is a discrete event simulator that has been widely used. However, SimJava has several scalability limitations that led the developers of CloudSim to implement a new discrete event management framework which is the CloudSim core simulation engine.

Although the above discrete event-based simulators are widely used in the last decades, they have lots of limitations and shortcomings causing failure to model and simulate cloud-hosted BDAAs. Take the most popular discrete event-based simulator Cloudsim as an example, CloudSim shows several weaknesses. First, it is built on top

of a grid computing environment which puts limitations on the infrastructures it can simulate. Second, it only includes a basic and simplified network model and a limited workload generator. Third, CloudSim is only able to model different CC features and components such as data centers, hosts and VMs on an elementary level without the capabilities of modeling BD processing paradigm (for example MapReduce). Fourth, it lacks SLA components to handle the processing of SLA-aware BDAAs problems. To the best of our knowledge, there are only a few DES simulators in particular on modeling big data processing model such as MapReduce. They are introduced as follows.

- MRPerf [306]: A design tool for MapReduce infrastructure and can help in designing new high-performance MapReduce setup, and in optimizing existing ones. However, it cannot simulate the complete behavior of a Hadoop framework. Also, [338] claimed that accurate results for jobs of different type of algorithms or different cluster configurations could not be generated, based on the testing they performed on the MRPerf code.

- Mumak [339]: An open source Apache's MapReduce simulator which uses data from real experiments to estimate the completion time for Map and Reduce tasks with different scheduling algorithms. In cases where data from real experiments do not exist, Mumak cannot estimate completion time for Map and Reduce tasks.

- SimMR [340]: Was developed in HP lab. It can replay execution traces of real workloads collected in Hadoop clusters (as well as synthetic traces based on statistical properties of workloads) for evaluating different resource allocation and scheduling ideas in MapReduce environments.

- MRSim [338]: A discrete event based MapReduce simulator. It can simulate different type of MapReduce BDAAs with the ability to study with good accuracy the effect of dozens of job configuration parameters on job performance. However, it was modeled and simulated using SimJava discrete event engine that has inherent weakness such as increased kernel complexity [341] and lack of support of some advanced operations [332]. Because of this, the SimJava layer has been removed from Cloudsim 2.0 onwards.

- MR-Cloudsim [342]: Was developed for simulating MapReduce processing model. However, MR-Cloudsim has several limitations as it only supports simple, single-state Map and Reduce computation. Further, it lacks support for network link modeling, which is a critical element affecting the performance of MapReduce BDAAs. Also, there is a lack of support for allowing the execution of multiple MapReduce BDAAs concurrently.

It is concluded that it is necessary to design and implement a new discrete event-based simulators simulator with the enhanced capabilities of efficiently modeling and simulating cloud-hosted BDAAs.

5.2 Modeling Requirements for Cloud-hosted BDAAs

Nowadays, the generation of cloud-hosted BDAAs combines multiple independent data analytics models, historical data repositories that are likely to be available across geographically distributed datacenters (both private and public). Typically, such BDAAs need to process large amounts of data by using parallel big data processing technologies such as MapReduce. MapReduce is a parallel programming paradigm initially proposed by Google to generate and process large datasets in clusters, which is becoming the de facto tool for BDAAs. Hence, we are going to analyze the modeling requirements for MapReduce-based BDAAs.

5.2.1 The MapReduce Model

MapReduce has been widely used as a predominant parallel processing model for large-scale dataset [141]. It is a programming model based on the divide and conquers paradigm that has efficiently solved problems of large datasets using large clusters of machines. Cloud computing (CC) is a suitable environment for processing and analyzing terabytes of data through utilization of many resources connected. Most cloud providers such as Amazon EC2, Microsoft Azure, and Google adopted MapReduce in their computing environments.

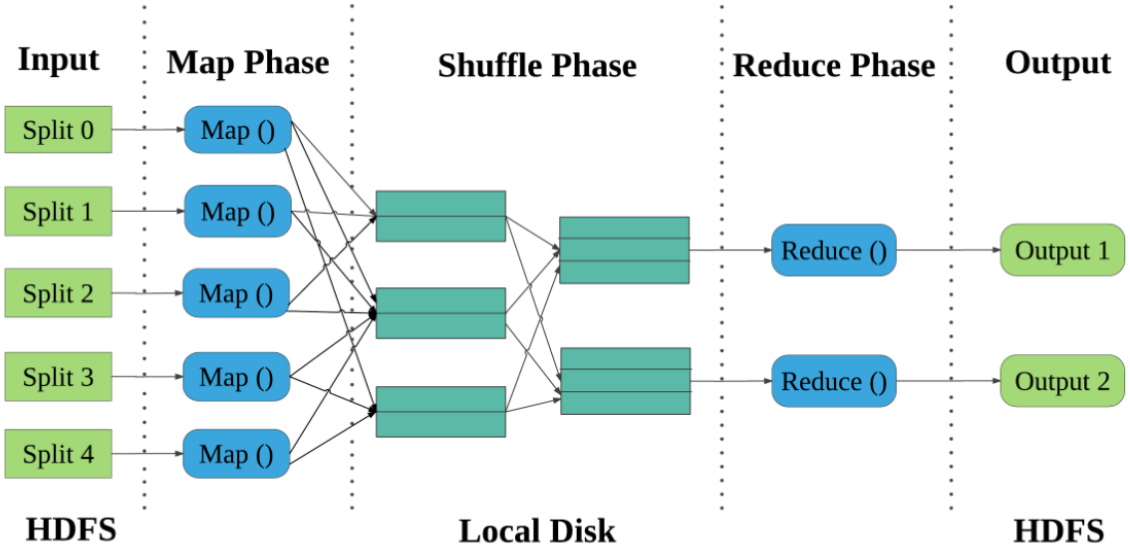


Figure 5.1: The workflow of MapReduce operation

MapReduce divides the job into tasks. There are two types of tasks including map tasks and reduce tasks. In the map tasks phase, an application reads input data and generates intermediate keys. These computations are generally done in parallel by subtasks, which are called map tasks. Then, in the reduce phase, another subtasks, called as reduce tasks, read intermediate keys and produce the program results. Map tasks perform filtering and sorting operations while reduce tasks perform summary operations. The user can specify map/reduce functions, and types of input/output. Figure 5.1 shows the workflow of MapReduce operation.

The complete execution process (execution of Map and Reduce tasks) is controlled by two types of entities called JobTracker and multiple TaskTrackers shown in Figure 5.2. For every job submitted for execution in the system, there is one JobTracker that resides on NameNode, and there are multiple Tasktrackers residing on DataNode. A job is divided into multiple tasks which are then run onto multiple data nodes in a cluster. It is the responsibility of the jobtracker to coordinate activities by scheduling tasks to run on different data nodes. Execution of individual task is then to look after by TaskTracker, which resides on every data node executing part of the job. TaskTracker's

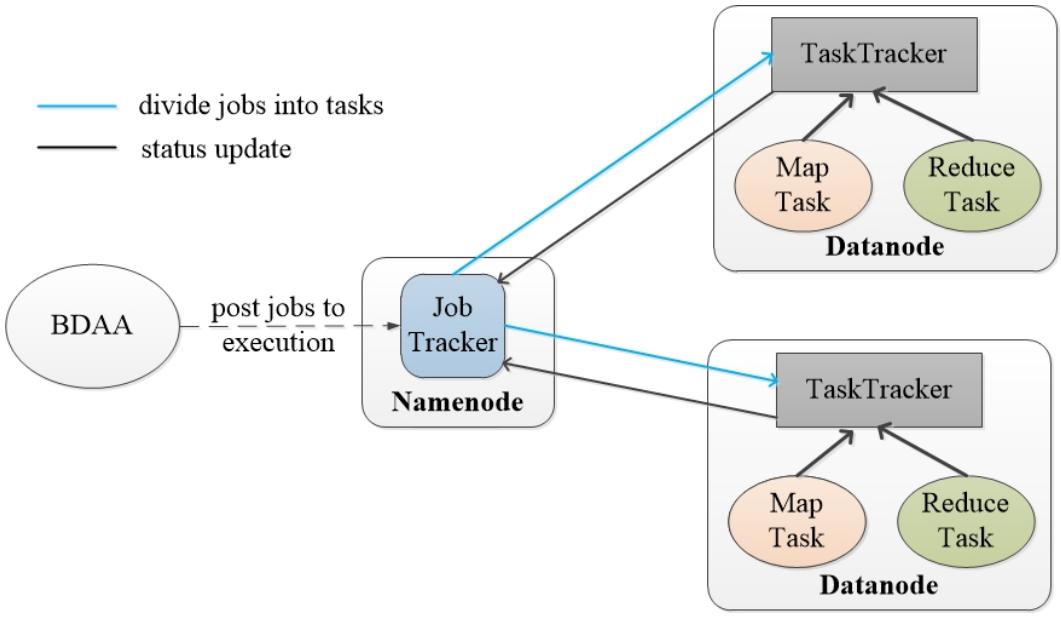


Figure 5.2: The MapReduce architecture in Hadoop

responsibility is to send the progress report to the JobTracker. Also, Tasktracker periodically sends 'heartbeat' signal to notify JobTracker current state of the system. Thus Jobtracker keeps track of the overall progress of each job. In the event of task failure, the JobTracker can reschedule it on a different TaskTracker.

5.2.2 BDSaaS-level Requirements

A cloud-hosted BDAA generally processes large datasets stored in clouds. The simulation tool should thus allow modeling of different applications depending on which big data processing framework at BDPaaS layer such as MapReduce will be used. For example, a MapReduce application may consist of one or more jobs with particular workloads, and specific SLA requirements such as deadline, budget, and so on. The proposed simulation tool must meet these application-level requirements, allowing researchers setting various application profile according to different needs.

5.2.3 BDPaaS-level Requirements

To support cloud-hosted BDAAs, big data processing technologies at BDPaaS layer plays a central role. Hence, it is mandatorily required for the proposed simulation tool to meet these platform-level requirements. Take the popular big data processing paradigm MapReduce as an example, the simulation tool must offer the capabilities that mimic how MapReduce works when processing a large amount of dataset. Also, it must allow to effectively model and simulate the execution of multiple jobs simultaneously in a scalable manner as it happens in the real world. The MapReduce working mechanism will be detailed in Section 5.3.

5.2.4 ClaaS-level Requirements

In order to support MapReduce programming model, cloud-hosted BDAAs need lots of VMs at the underlying cloud infrastructure layer. These VMs might have different size and contain different CPU, memory, and storage. Hence, a simulation tool must flexibly offer the capabilities of simulating these computing resources meeting different needs. Also, cloud-hosted BDAAs require different types of storage that are commonly ambient in cloud-based datacenters to store content from the various devices. Thus, a storage layer should be modeled to simulate the storage and retrieval of any amount of data, subject to the availability of network bandwidth. It is evident that accessing files in storages at run-time incurs additional delay for cloud-hosted BDAA execution. This is due to the latencies between the nodes and storages when transferring data files through cloud. Hence, the design of the network between hosts and storage is required to model the delay above.

5.3 The Design Decisions and Architecture of IoTSim

Our proposed simulation tool called IoTSim uses CloudSim as the underlying design platform. The reasons are manifold. Cloudsim as an discrete event-based simulator offers lots of features like support for modeling service brokers, resource provisioning,

application allocation policies, and simulation of large-scale cloud environments including datacenters, on a single computing machine. To be specific, Cloudsim supports modeling and creation of one or more VMs on a simulated node of a datacenter with different hardware configurations, cloud-based tasks and their mapping to suitable VMs. It also allows simulation of multiple datacenters to enable a study on federated and associated policies for migration of VMs for reliability and automatic scaling of applications. Also, Cloudsim helps in modeling applications having independent jobs, and design and analysis of different hardware configurations, VM provisioning, and scheduling policies. Thus, Cloudsim can pave the way for us to design and implement our tool to simulate BDAAs in cloud. In fact, Bashar [343] had done a critical evaluation of various simulators in cloud environment, and his study has concluded that Cloudsim is the best choice if research has to be done by using simulation software. All of these features enable us to choose Cloudsim as a preferable platform to design and implement a new simulator IoTSim with the purpose of simulating and analyzing cloud-hosted BDAAs.

On top of Cloudsim, we introduce many new enhancements to meet the modeling requirements. In the following subsections, we will details how we consider the design principles, architecture and fundamental classes of IoTSim by extending Cloudsim.

5.3.1 Design Principles

We consider several essential principles when designing our proposed simulation tool.

- Layer-based: This principle offers lots of advantages. Not only it is possible to provide both power and ease of use in a scalable framework, but also the architecture is clear and loose-coupled, which offers the flexibilities for future researchers or developers to extend on top of it without fundamentally changing the structure.

- Entity-oriented: Entity refers to something which can individually and independently exist. It can send messages to other entities, and process received messages as well as trigger and handle events. Each entity is initiated at the beginning and shutdown at the end during the simulation. When using MapReduce to process

the applications, lots of entities like JobTracker, Task Tracker(s), MapTask(s) and ReduceTask(s) are demanded. It is necessary to follow this entity-oriented principle to design and implement a simulation tool.

- Event-driven: Event represents a simulation event which is passed between entities in the simulation. Each event carries all the related information about an event between two or more entities such as event type, init time, time at which the event should occur, time at which the event should be delivered to its destination entity, the source entity and the destination entity as well as the data that has to be passed to the destination entity. According to the working mechanism of MapReduce, lots of communications in the form of heartbeat between different components/entities are generated. These entities work tightly by sending and receiving messages in order to trigger or handle events.

5.3.2 Specific Design Considerations

5.3.2.1 BDSaaS-level Design

Most simulation tools designed for cloud environment generally offer limited support for modeling execution of parallel and distributed applications. For example, Cloudsim allows modeling an application service or a cloud task by using a programming structure called "Cloudlet" class, which only represents single and atomic computation needs. This approach is not appropriate in cloud-hosted BDAAs scenario. Instead, we introduce two types of Cloudlet including MapCloudlet and ReduceCloudlet. The details of the two types of Cloudlets will be described thereafter. Both of them inherits from Cloudlet class in Cloudsim and has their own specific attributes. ReduceCloudlet always runs after MapCloudlet over the same input as it does in the real world.

5.3.2.2 BDPaaS-level Design

We consider some important features to meet platform-level requirements. In Cloudsim, when Cloudlets come in, they are simply submitted to a cloud datacenter via a broker. However, it does not suit for submitting MapReduce job in the same way due

to the complexity of MapReduce workflow. Since JobTracker and TaskTracker play an integral part during the MapReduce processing, the proposed simulation tool needs to entirely model and implement them as well as other notable features that MapReduce model has as it does in real world. Such features include: (i) lots of communication and interaction between entities during the execution; (ii) every separate map task output has one corresponding reduce operation; (iii) reduce operation has to come after map operation over the corresponding intermediate input; (iv) multiple MapReduce jobs can be submitted and run concurrently in a shared cloud-hosted datacenters.

5.3.2.3 CIaaS-level Design

A significant amount of data from various sources are ingested and stored at CIaaS layer. In the runtime environment, a map instance (mapper) operates in each slave node. It copies data which is saved in the above data storage component to its local hard disk. When the data is copied and saved in the local hard disk, mapper starts processing the allocated map task by TaskTracker. After that, the intermediate output will be generated and are associated with the reduce instance. The paired reducer reads the intermediate output and starts processing the allocated reduce tasks. The final output will be written into the storage component. Therefore, there is two typical network delay incurred that affects the performance of map or reduce task(s). In this scenario, the network and storage model must be represented in the proposed simulation tool. One of the feasible methods is to calculate the network consumption when copying data from storage component by mapper or copying intermediate output from local disk by the reducer. It is enough to guarantee the accuracy of the simulation tool while calculating and presenting the network delay incurred during the processing.

5.3.3 The Architecture of IoTSim

In this section, we propose a layer-based architecture of IoTSim shown in Figure 5.3. The detailed entities/classes design and functionality of our proposed simulation tool's components will be discussed thereafter.

- Cloudsim Core Simulation Engine Layer: Located at the bottommost layer. It is a simulation engine that supports several core functionalities, such as queuing and processing of events, the creation of cloud system entities (services, host, datacenter, broker, and VMs), communication between components, and management of the simulation clock.

- Cloudsim Simulation Layer: This layer provides support for modeling and simulation of virtualized cloud-based datacentre environments including dedicated management interfaces for VMs, memory, storage, and bandwidth. The fundamental issues such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state are handled by this layer. This layer consists of several sub-layers that model the core elements of cloud. The bottommost sub layers model datacenters, cloud coordinator, and network topology. The VM Services and Cloud Services provide the functionality to design VM resource and application scheduling algorithms.

- Big Data Platform Layer: It includes two components. MapReduce component is to support BDAAs where a batch-oriented data processing paradigm is required while Hadoop distributed file system (HDFS) components offer the abstraction of data storage. Depending on which cloud-hosted BDAAs customers will use, it can support data processing using MapReduce programming paradigm running on HDFS. Take the MapReduce applications as an example, a MapReduce model needs to be fully implemented here where a set of new classes or entities such as JobTracker, TaskTracker, Mapper and Reducer work as does the real Hadoop and a series of events occur in some specific order to finish a Map/Reduce process. Due to the aforementioned limitation of Cloudsim, the BD Platform Layer is the primary extension of Cloudsim and plays an integral role in support of modeling, simulation, and analysis of cloud-hosted BDAAs.

- User Code Layer: The top-most layer which exposes basic entities for hosts (number of machines, their specification and so on), cloud-hosted BDAAs configurations (application workload and their SLA requirements), VMs, number of users and

their application types and so on. This layer helps users to define their simulation scenarios and configurations for validating their algorithms.

5.4 The Development of Fundamental Classes and Their Correlations

This section provides fine-grained details regarding the fundamental classes/entities of IoTSim during the development phrases. We firstly build a Cloudsim-based development environment using an IDE tool (e.g., Eclipse) installed in a Linux machine, which allows us to develop our proposed simulator.

Since IoTSim extends Cloudsim, the development consists of two main parts including modification and addition. Modifications are done on the original Cloudsim code including Datacenter, CloudTag, and DatacenterBroker. In terms of additions, we developed many new classes and the logic between them based on the aforementioned design decisions.

The detailed diagram of these classes and their correlation is shown in Figure 5.4. The list of classes are:

- BDBroker: This new data center broker is responsible for mediating negotiations between providers and customers with respect to allocating appropriate resources to MapReduce-based BDAAs to meet SLA requirements and to manage providers' resources. The new datacenter broker includes the capability of running dynamic MapReduce processing simulations thereby removing the burden of statically configuring the whole simulation scenario before starting the simulation.

- MapCloudlet: This class inherits Cloudlet. It models the atomic map task's information and behaviors such as input dataset size, output dataset size, and utilization model. It will be submitted by BDBroker to the runtime environment and then executed in a designated VM.

- ReduceCloudlet: This class inherits Cloudlet. It models the atomic reduce task's information and behaviors such as input dataset size, output dataset size, and

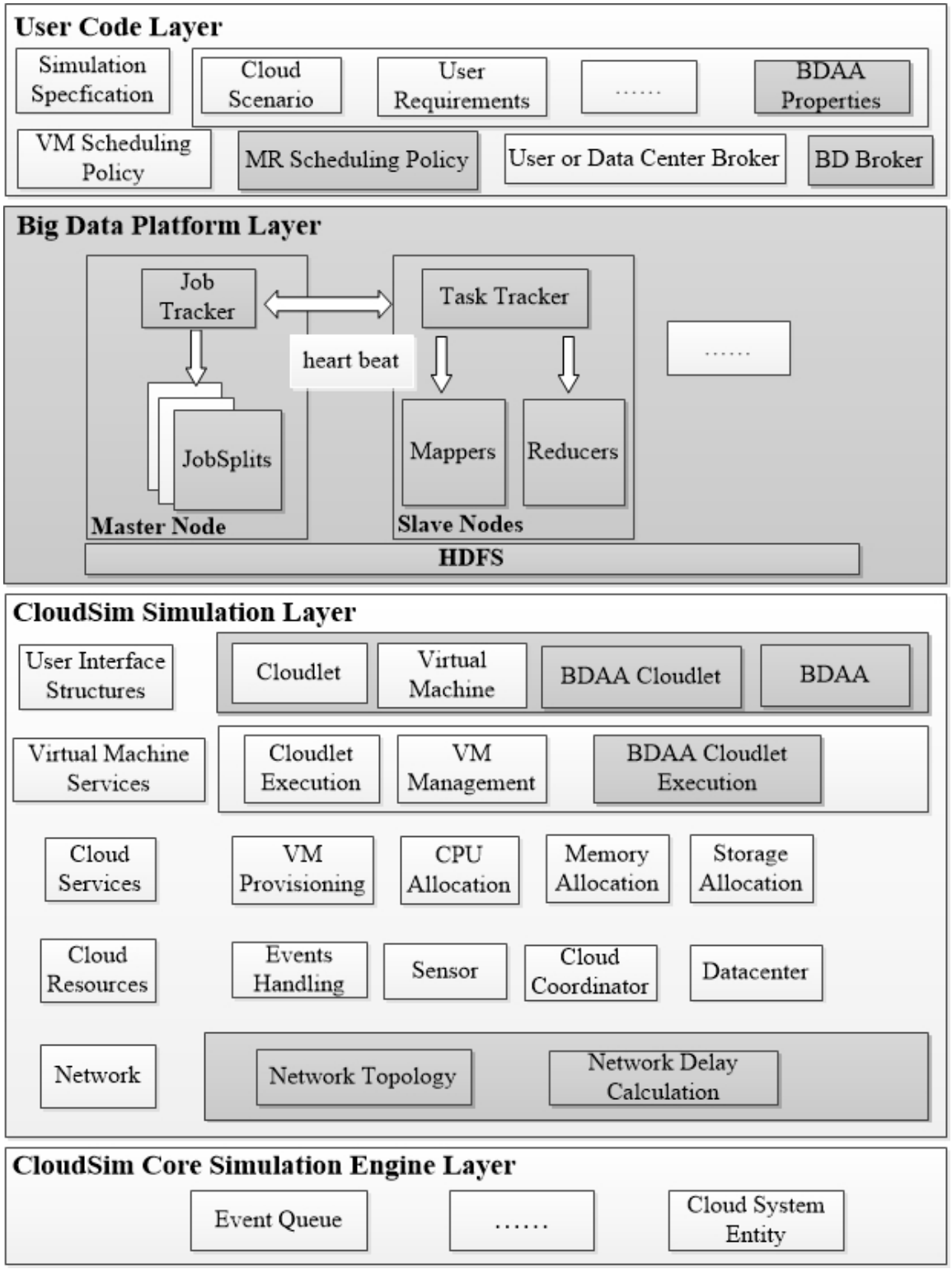


Figure 5.3: The proposed layered architecture of IoTSim

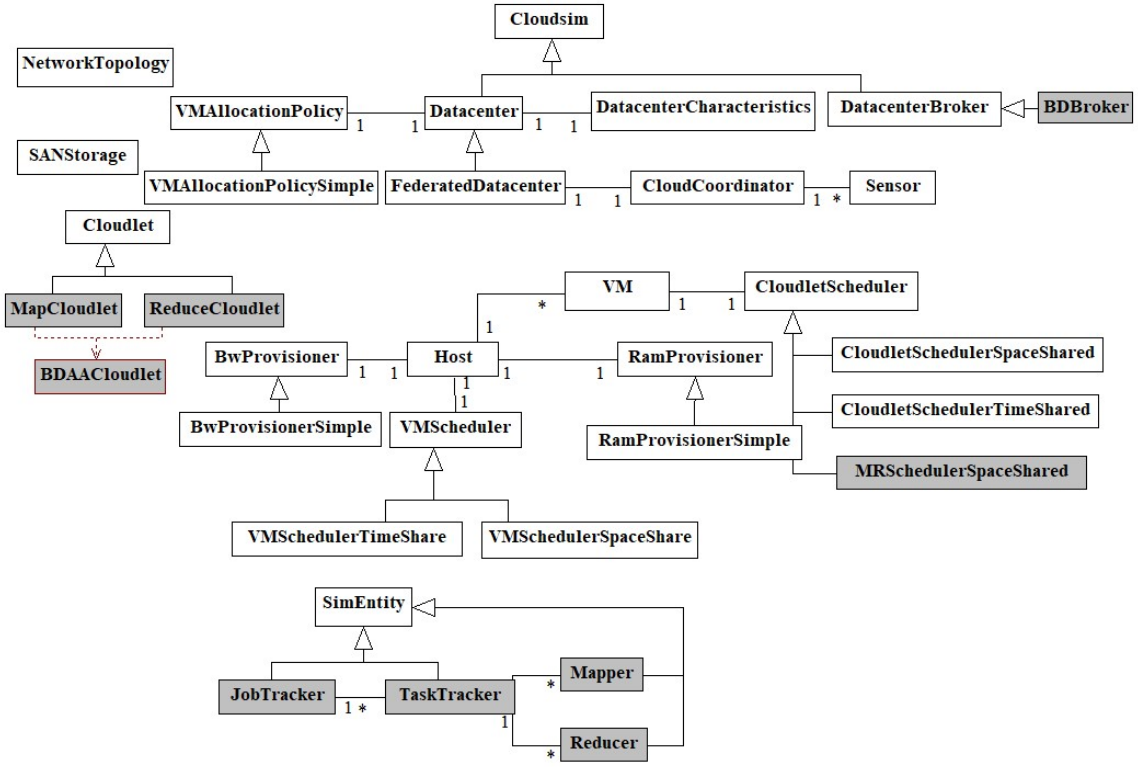


Figure 5.4: Class diagram of fundamental entities

utilization model. It will be submitted by BDBroker to the runtime environment and executed in a designated VM.

- BDAACloudlet: This class contains instances of all mappers and reducers that belong to the same user. Also, it contains information about the status and data locations during the execution of the mappers and reducers.

- JobTracker: An entity that receives the jobs summited by a user, gets the data from storage, splits the jobs according to user requirements and schedules them to the TaskTracker entity for execution. In the runtime environment, the JobTracker communicates with TaskTracker(s) by receiving status updates from them to track their progress. If JobTracker is aware that all Mappers have successfully finished their map tasks, then it will communicate with TaskTracker to launch the corresponding Reducer entity where reduce task will be processed. It produces

the intermediate output results after each Mapper finishes its work and provides input for Reducer.

- TaskTracker: This class represents an entity receiving processing requests from the JobTracker. Its primary responsibility is to track the execution of map and reduce tasks running locally on its hosted node and to report the status updates of each map and reduce tasks to the JobTracker. TaskTracker manages the processing resources on each slave node in the form of processing slots — the slots defined for map tasks and reduce tasks. It schedules the split map tasks and reduce tasks to the corresponding Mapper and Reducer, and monitors the status updates of them.

- Mapper: This class denotes an entity that receives the request of TaskTracker and communicates with BDBroker to submit the corresponding map tasks which will be executed in a datacenter. It frequently reports the status of map tasks to TaskTracker.

- Reducer: An entity that receives the request of TaskTracker and communicates with BDBroker to submit the corresponding reduce tasks to be executed in a datacenter. It frequently reports the status of reduce tasks to TaskTracker.

- MRSchedulerSpaceShared: This class extends an abstract class CloudletScheduler. It represents the behavior's policy between the Mapper and Reducer. For example, it determines the sharing policy of processing power among Mappers and Reducers in a VM.

It is worth noting that we develop IoTSim by extending and enhancing Cloudsim with new functionality so that it can support executing multiple CloudletLists sequentially. The work originated from a functionality limitation existing in current Cloudsim. Internally, the broker has a very simple operation and has a single CloudletList, which means if you submitted multiple CloudletLists to the broker, they are always merged to a single list, and they are handled as if only one submission were made. However, this is not the case for the MapReduce framework in the real world where the reduce operation

can only start after the corresponding map operation. In order to solve this problem, we developed a new broker called BDBroker which inherits the original broker in Cloudsim. This new broker can accept multiple Cloudlets and execute them concurrently. By this, the new broker can guarantee execution of the reduce task after corresponding map task has ended.

5.5 An Example of Simulation Using IoTSim

Based on our development, each simulation in IoTSim follows a set of specific steps in java code from configuration to simulation. We give an example showing how to create 10 datacenters with 5 hosts each and run MapReduce jobs of 20 concurrent users on them. Each user's MapReduce job has different workload and is generated by discrete uniform distributions. The simulation steps are shown in Figure 5.5.

1. Setting the number of users for a current simulation (e.g., 20). This is where the number of concurrent users can be varied
2. Initializing the simulation by instantiating the common variables (current time, trace flag, number of users)
3. Creating Cloud Information Service (CIS) instance
4. Creating data center instance (e.g., 10) and then registering it with CIS.
5. Creating physical machines (hosts) with their characteristics. Each datacenter contains 5 hosts. This step and step 4 can be easily done through the function of createDatacenter() in our code
6. Creating BDBroker instance
7. Creating JobTracker and TaskTracker
8. Creating VMs with their characteristics. A VM can be created in the way of new Vm() in our code
9. Submitting VMs to BDBroker

10. Creating multiple user-defined MapReduce-based BDAA cloudlets and specifying their characteristics from the user input module such as reading user-specified job information from JSON or XML format files
11. Start Simulation once there is an event to be executed
12. runClockTick() function works checking each SimEntity object and its state
13. JobTracker splits BDAA Cloudlet into some chunks and schedules them to some TaskTrackers for execution
14. TaskTrackers allocate the split job into the corresponding Mapper and Reducer
15. Mapper and Reducer submit the corresponding map tasks and reduce tasks to the VMs where they are executed. The status of the map tasks and reduce tasks are reported to TaskTracker
16. Submitting MapCloudlet and ReduceCloudlet to BDBroker
17. MapCloudlet and ReduceCloudlet return the result
18. Sending a call to destroy VMs once there is no event to be executed
19. Evoking StopSimulation( ) function to stop simulation
20. Printing the results of the simulation

In the runtime, BDBroker works linking each object. When the linking ends, the simulator takes action on captured event time. An event occurs when Cloudsim creates, executes, and terminates each object such as datacenter, BDBroker, VM, JobTracker, and TaskTracker. runClockTick() function checks each SimEntity object and its state. If the state is runnable, each SimEntity object classifies its own operable events. Each entity checks simulating tag and operates each request. Each object has one various tag. They consist of entity creation, acknowledge, characteristic setting, event pause, move, submit, migration, and termination. At the event time that the cloudlet process is submitted, the simulator calculates all submitted cloudlet's processing time. During

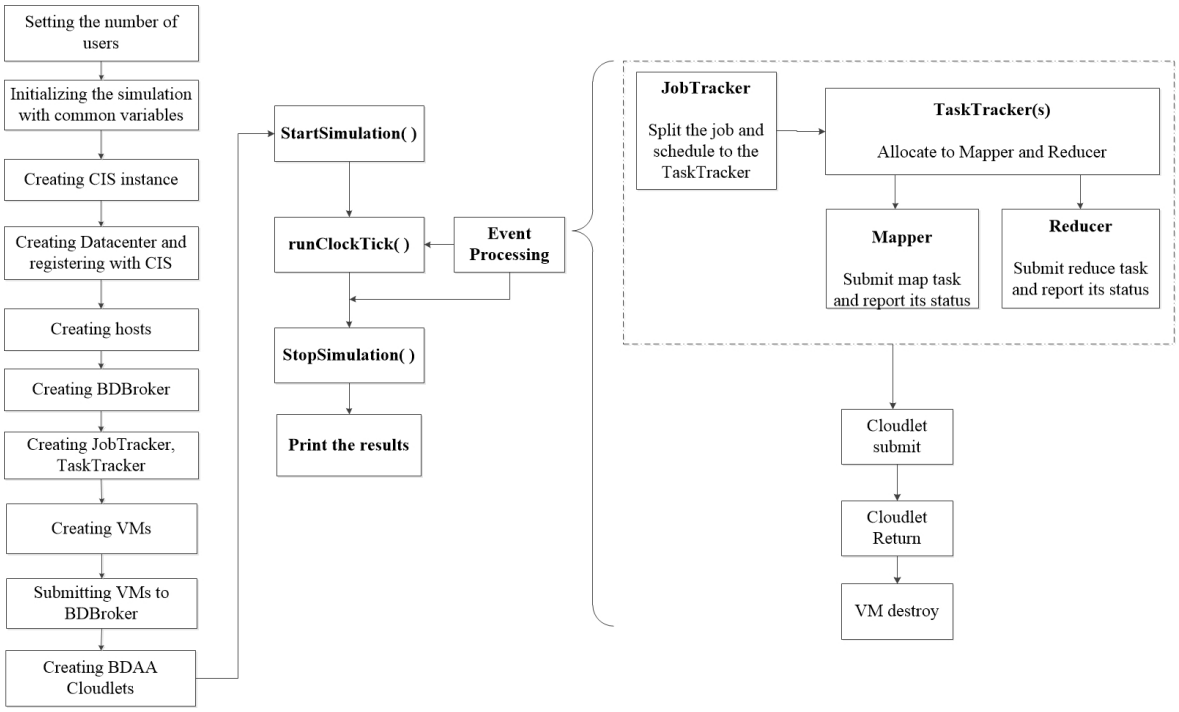


Figure 5.5: The basic steps in the simulation of IoTSim

the event processing, a new event may be created. When new event is created, send() or sendNow() function is called. These functions notify that one event time is created. When all event time is over, then simulating ends and the simulation result is reported. The simulation reports consist of each MapReduce-based BDAA's information (e.g., application name, number of map tasks, number of reduce tasks, job length), status, executed in which datacenter and which VM, and processing result (e.g., job id, VM id, start time, execution time, finish time, VM computation cost).

During the simulation, users might encounter run-time exceptions. For example, insufficient resources (e.g., CPU cores, memory size or disk size) in datacenters to process a MapReduce job. In this, case, a report will be generated for users to readily troubleshoot where the exceptions come from our code. However, we assume that all MapReduce jobs can be successfully executed on the prerequisite that there are enough resources provided by datacenters. Hence there is a limitation that IoTSim is not able to handle fault tolerance of MapReduce.

5.6 Evaluation

In this section, we evaluate the efficacy of our proposed simulation tool (i.e., IoTSim) by conducting extensive experiments to observe how the independent variables impact the dependent variables (see Section 5.8.2 and 5.8.3).

5.6.1 Experiment Setup

We install Cloudsim firstly and then deploy our simulator on top of it. To imitate the behaviour of a real MapReduce-based BDAA (e.g., Google inventory system), we consider multiple jobs with different user-specified parameters (e.g., Job Length, Data Size etc.,) submitted to many VMs (having different parameters in terms of MIPS, pesNumber etc.,) that are hosted in Datacenter(s) in our simulated environment. The incoming jobs are processed using the MapReduce processing engine which produces the corresponding output once the jobs are finished. The output contains the relevant job information including job id, job type, job length, job size, start time, execution time, finish time and the VM (denotes by unique ID) in which map task or reduce task is processed.

We consider two cases in our experiments:

- Without Network Delay: In this case, JobTracker splits the MapReduce jobs according to user requirements and then schedules them to the TaskTracker node for execution immediately when it receives the jobs summited by the user. Also, when all the Mappers finish the map tasks successfully, the corresponding reduce tasks begin to execute immediately. This scenario means no network delay is incurred during the whole period that the job is running in the simulated environment.

- Network Delay: In this case, JobTracker firstly gets the data from storage for each MapReduce job when the simulation begins, this causes the first delay that job starts after the simulation clock. By subtracting the start time of the map task and the start time of simulation clock time, this delay can be captured. When all Mappers finish the map tasks, each Mapper will produce an intermediate output.

Then, the corresponding Reducer begins to work based on the intermediate output. Apparently, this causes the second delay. This delay can also be captured by subtracting the start time of reduce task and the finish time of the corresponding map task.

Most importantly, we define a series of key independent variables and dependent variables in our experiments. Independent variable includes datacenter parameters, VM parameters, the number of VM, job parameters, and the number of MapReduce tasks while dependent variable includes average execution time, maximum execution time, minimum execution time, makespan, delay time, total VM cost, and total network cost. These independent variables are the main factors that affect the above dependent variables. They will be detailed in the following subsections.

5.6.1.1 Application Workload Generation

On the user's side, a request for executing MapReduce jobs contains details of application workload characteristic. Users can generate parameters per job in workload using discrete uniform distributions. In our proposed simulation tool, we assume that all jobs arrive at the same time and ready to be processed. Hence, there are limitations in our proposed simulation tool that users can not specify probability distribution for job arrival rate, or user distribution over a time frame.

5.6.2 Independent Variables

5.6.2.1 Datacenter Configuration

Datacenter models the physical hardware that is offered by CIaaS provider. It encapsulates a set of computing hosts that can either be homogeneous or heterogeneous with their hardware configurations (memory, CPU, and storage) where a specific number of hosts and VMs are generated and run. Table 5.1 lists the datacenter parameters that are going to be consistent throughout the entire evaluation.

Table 5.1: Datacenter configuration

Datacenter ID	**A unique ID automatically generated**
System architect	x86
Operation System	Linux
pesNumber // number of CPUs in a host	500
RAM //host memory (GB)	20
Storage //host storage (GB)	1024
Bandwidth //amount of bandwidth (Mbps)	1000
MIPS //millions of instructions per second	1000

In Cloudsim, there are a few other parameters including OS, system architect, etc., to be defined for initialization. They are not factors that can affect the aforementioned dependent variables. Hence, such parameters are excluded from our experiments.

5.6.2.2 Host Parameters

Each datacenter has identical physical hosts with properties shown in Table 5.2.

Table 5.2: Host parameters

Host ID	**A unique ID automatically generated**
Storage Capacity (MB)	1024
Number of CPUs	2
Cores per CPU	4
MIPS for each core	1024
Memory capacity // (GB)	2
Bandwidth //amount of bandwidth (Mbps)	10
MIPS //millions of instructions per second	1000

5.6.2.3 VM Parameters

VM contains the following characteristics: processor, memory, storage size, bandwidth, and MIPS. It should be submitted to the broker before simulation starts. VM parameters are monitored in Cloudsim, which means each sum of VM parameter must be less than the corresponding datacenter configuration. For simplicity, we define three types of VM (Small, Medium, and Large) that are compatible with typical computing infrastructure in Amazon EC2. The configuration of three types of VMs is listed in Table 5.3.

Table 5.4: Job parameters

Job Type	**Small**	**Medium**	**Big**
Job Length // the length (expressed in millions of instructions) of this job to be executed in a datacenter	362880	725760	1451520
Data Size // the data size (in MB) of this job before submitting to a datacenter	200000	400000	800000

Table 5.3: VM parameters

VM Type	**Small**	**Medium**	**Large**
Storage Size //amount of storage (MB)	80	160	320
Memory //VM memory (MB)	512	1024	2048
MIPS for each core //millions of instructions per second	1000	2000	4000
Bandwidth //amount of bandwidth	1000	1000	1000
pesNumber //number of CPUs in a VM	2	4	8
VMCost per seconds // the cost of processing in VM	0.350	0.700	1.400

5.6.2.4 The Number of VMs

Further to the above VM Configuration itself, we will also specify the number of VMs. This independent variable refers to the quantity of VM in a datacenter. VM, which is hosted in the host, is the atomic computing unit where the jobs will be processed. In our experiment, we will change the number of VM as needed. For the sake of simplicity, we set the number of VMs to be 3, 6 or 9. It is also feasible and flexible to set a considerable number of VMs in IoTSim. In this case, the capacity of the datacenter hosting these VMs should be adjusted accordingly to avoid insufficient physical resources error.

5.6.2.5 Job Parameters

In order to demonstrate the effect of resource variances when multiple BDAAs requests arrive with different parameters, we simulated a series of BDAAs requests arriving in cloud. Each request has a set of 10 random input requests coming in with different job parameters, for example, a pre-assigned instruction length and data transfer overhead that it needs to undertake during its life cycle. We define three types of job parameters in Table 5.4.

5.6.2.6 The Number of MapReduce Tasks

This represents the specific number of map tasks and reduce tasks. In IoTSim, when a datacenter receives a job with user-specified job length and data size, the number of map tasks and reduce tasks in each job should also be considered throughout the experiment. For example, M1R1 means there is one map task, and one reduce task for this job. Similarly, M20R1 means there are twenty map tasks, and one reduce task for this job. Since a job might have an endless different value of the number of map tasks or reduce tasks, for simplicity, we consider a job has up to twenty map tasks and one reduce task.

5.6.3 Dependent Variables

Dependent variables are crucial factors to evaluate the efficacy of IoTSim when modelling and simulating cloud-hosted BDAAs. A dependent variable changes accordingly as the aforementioned independent variables change. For example, given the same cloud-hosted BDAA job, if we increase the number of VMs or provide higher VM type, the makespan and execution time of map or reduce tasks may change because increased or more high-profile VMs resources can be utilized by map or reduce tasks to process this job. For clarity and quick reference, we firstly define the notations in Table 5.5 to clearly describe dependent variables.

5.6.3.1 Average Execution Time

It refers to the average execution time of a MapReduce job J_i. Its value is given by

$$Average\,Execution\,Time = \frac{\sum_{i=1}^{N_{map}} ET_{map}^{i}}{N_{map}} + \frac{\sum_{j=1}^{N_{reduce}} ET_{reduce}^{j}}{N_{reduce}}$$

5.6.3.2 Maximum Execution Time

It means the maximum execution time of MapReduce job J_i. It is calculated by

$$Maximum\,Execution\,Time = \max_{1 \le i \le N_{map}} ET_{map}^{i} + \max_{1 \le j \le N_{reduce}} ET_{reduce}^{j}$$

Table 5.5: Notation used in description of dependent variables

Symbols	Description
J	a set of MapReduce jobs or workloads
J_i	a MapReduce job or workload instance in J. For generality, each J_i has different user-specified attributes (e.g., N_{map}, N_{reduce} etc.,)
N_{map}	the number of map tasks in job J_i
N_{reduce}	the number of reduce tasks in job J_i
DT^i_{map}	the delay time of map task i, $i \in [1, N_{map}]$
DT^i_{reduce}	the delay time of reduce task j, $j \in [1, N_{reduce}]$
ST^i_{map}	the start time of map task i, $i \in [1, N_{map}]$
ST^i_{reduce}	the start time of reduce task j, $j \in [1, N_{reduce}]$
ET^i_{map}	the execution time of map task i, $i \in [1, N_{map}]$
ET^i_{reduce}	the execution time of reduce task j, $j \in [1, N_{reduce}]$
FT^i_{map}	the finish time of map task i, $i \in [1, N_{map}]$
FT^i_{reduce}	the finish time of reduce task j, $j \in [1, N_{reduce}]$
N_{vm}	the number of VMs
C_{vm}	the computing cost per hour incurred when VM is processing map tasks or reduce tasks (Unit: dollars)
$C_{network}$	the network cost per hour incurred when transferring data (Unit: dollars)

5.6.3.3 Minimum Execution Time

It represents the minimum execution time of MapReduce Job J_i. Its value is given by

$$Minimum\,Execution\,Time = \min_{1 \le i \le N_{map}} ET^i_{map} + \min_{1 \le j \le N_{reduce}} ET^j_{reduce}$$

5.6.3.4 Makespan

It is the period of MapReduce job J_i from start to finish. It is determined by the latest completed reduce task and calculated by

$$Makespan = \max_{1 \le i \le N_{map}} (DT^i_{map} + ST^i_{map} + ET^i_{map}) + \max_{1 \le j \le N_{reduce}} (DT^j_{reduce} + ST^j_{reduce} + ET^j_{reduce})$$

5.6.3.5 Total Delay Time

The total delay time consists of the delay time of map task i (i.e.,DT^i_{map}) and the delay time of reduce task j (i.e.,DT^j_{reduce}). We assume a map task is executed

immediately after an amount of delay and a reduce task is executed when the latest map task finishes. It is calculated by

$$Total\,Delay\,Time = FT^{i}_{map} - ET^{i}_{map} + ST^{j}_{reduce} - \max_{1 \le i \le N_{map}} EDT^{i}_{map}$$

5.6.3.6 Total VM Computation Cost

It refers to the CPU computation cost (Unit: dollars) incurred when a VM processes a Cloudlet. It is calculated by

$$Total\,VM\,Computation\,Cost = (\sum_{i=1}^{N_{map}} ET^{i}_{map} + \sum_{j=1}^{N_{reduce}} ET^{j}_{reduce}) * C_{vm}$$

5.6.3.7 Total Network Cost

This dependent variable means the network cost (unit: dollars) incurred when job task gets data from storage and reduce task gets data from the intermediate output of the map task. It is calculated by

$$Total\,Network\,Cost = (\sum_{i=1}^{N_{map}} DT^{i}_{map} + \sum_{j=1}^{N_{reduce}} DT^{j}_{reduce}) * C_{network}$$

5.6.4 Evaluation Scenarios

To understand the behavior and performance of our proposed simulation tool, we consider the following four types of scenarios. In each scenario, we change an independent variable, while keeping others constant. Hence, we can observe the result and perform appropriate analysis on how the dependent variable is impacted by the independent variable.

- Variation of the number of MapReduce tasks
- Variation of the number of VMs
- Variation of the types of VMs
- Variation of the job types

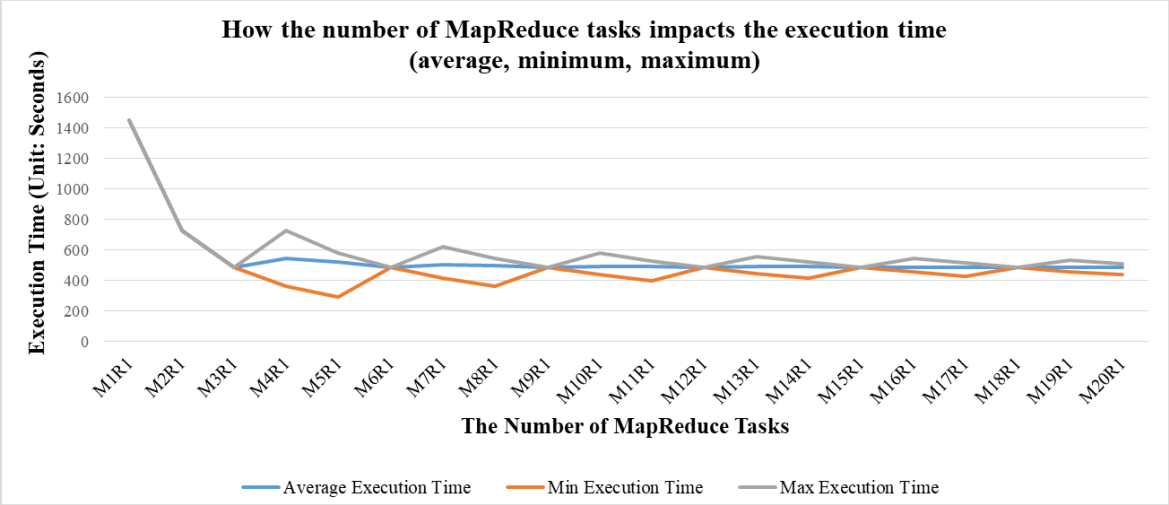
5.7 Experiment Results and Analysis

5.7.1 Variation of the Number of MapReduce Tasks

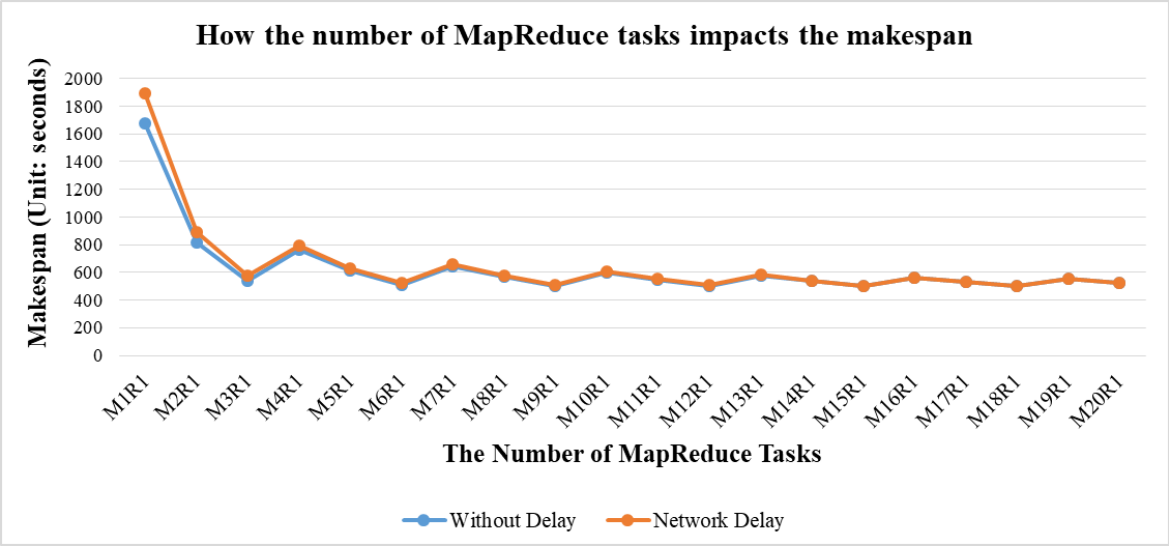
In this scenario, we set constant job parameters (e.g., Small Job), VM parameters (Small VM) and the number of VMs (e.g., 3 VMs), while varying the number of MapReduce tasks (e.g., from M1R1 to M20R1) for a user-specified job. We then start simulation in either without network delay or network delay case respectively. After the simulation successfully finishes, we collect the result of $Average\,Execution\,Time$, $Max\,Execution\,Time$, $Min\,Execution\,Time$, $Makespan$ and $VM\,Computing\,Cost$ and $Network\,Cost$ (only applicable for Network Delay case) based on the equations in Section 5.8.3. Figure 5.6 clearly shows how the number of MapReduce tasks impact the execution time (average, minimum and maximum) and makespan.

According to Figure 5.6(a), generally speaking, Execution Time (Average, Max, and Min) fluctuates as the number of MapReduce tasks increases. It is observed that when the number of map tasks is smaller than the number of VM, the average, maximum, and minimum execution time are identical because the datacenter provides redundant VMs that the map tasks of the job needs, which means some VMs are idle in this case. Also, it is seen that the average, maximum, and minimum execution time decrease rapidly when the number of MapReduce tasks increases. This is because more map tasks are generated and executed concurrently in the datacenter, which brings more time efficiency. However, if the number of map task is higher than the number of VM, the average, maximum, and minimum execution time become flattened, and the discrepancy between them become narrowed as the number of MapReduce tasks increases. The reason is that in this case, VMs become the bottleneck and many map tasks compete for the limited VMs, which results in that the effect of reducing execution time by increasing the number of MapReduce tasks becomes insignificant.

Figure 5.6(b) shows how the number of MapReduce tasks impact the makespan in two scenarios (network delay and without network delay). It is observed that the makespan in the scenario of network delay is slightly larger than the makespan in the scenario of without network delay and the disparity between them becomes narrowed as



(a) Execution time (Average, Max, Min)



(b) Makespan

Figure 5.6: How the number of MapReduce tasks impact execution time and makespan: (a) execution time (Average, Max, Min); (b) Makespan

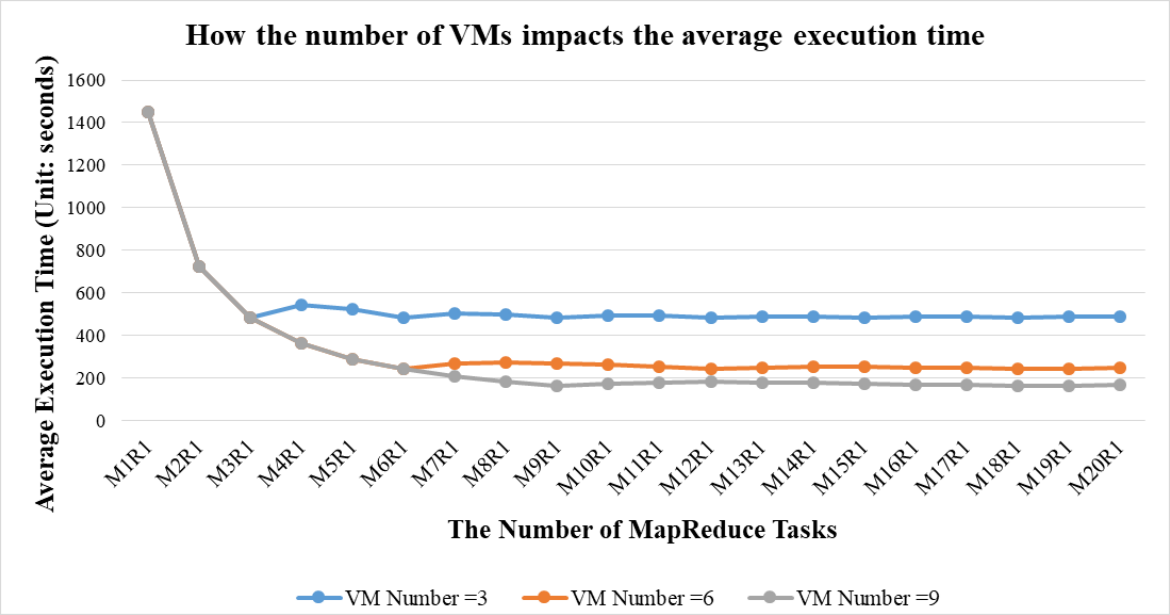


Figure 5.7: How the number of VMs impacts the average execution time

the number of MapReduce tasks increases. This is because in the scenario of network delay, copying data from storage and transferring intermediate output generated by map tasks incur an amount of delay that debilitates over the number of MapReduce tasks.

5.7.2 Variation of the Number of VMs

In this scenario, we set constant job parameters (e.g., Small Job), VM parameters (e.g., Small VM) and the number of MapReduce tasks (e.g., from M1R1 to M20R1) while varying the number of VMs. Since the number of VMs might have an endless variety values, without loss of generality, we set the number of VMs to 3, 6, and 9 respectively. The results in this scenario are presented in Figure 5.7 and Figure 5.8.

It is found from Figure 5.7 that when the number of map tasks is smaller than the number of VMs, their average execution time are equal. Afterward, the chart shows more VMs can apparently reduce the average execution time. To be precise, when the number of VMs increases from 3 to 6, the average execution time is roughly reduced

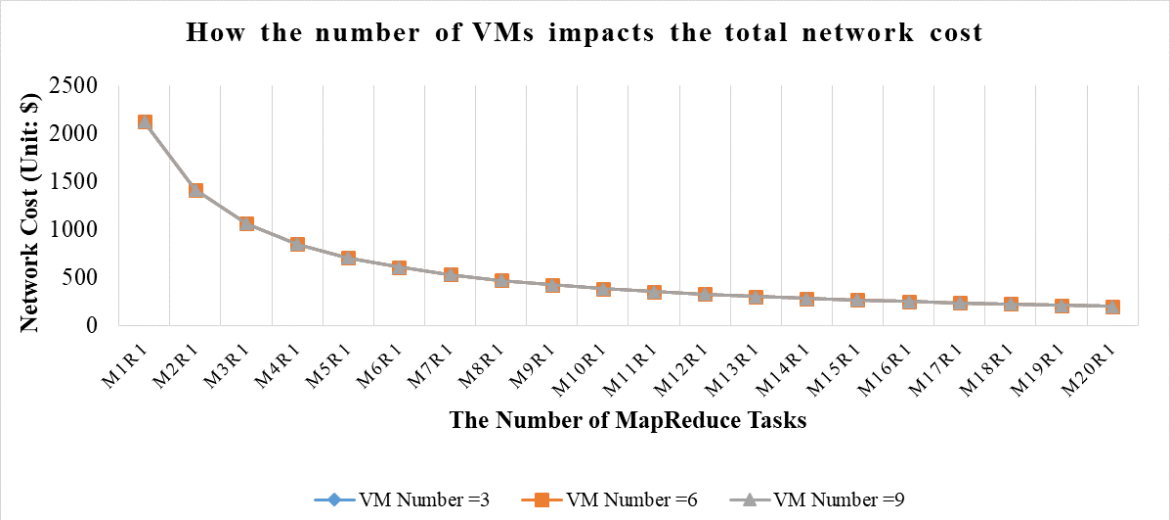


Figure 5.8: How the number of VMs impacts the total network cost

by 40% and 50% if it further increases to 9. This is because more VM resources are available for processing the MapReduce job.

Figure 5.8 presents how the number of VMs impacts the total network cost. Interesting, the network cost stays invariant when the number of VMs changes. This is because, given the same job, the data size is identical, which results in the same network delay.

5.7.3 Variation of the Types of VMs

In this scenario, we set constant job parameters (e.g., Small Job), the number of VMs (e.g., 3 VMs) while giving different VM types (e.g., from Small VM to Large VM). The result of this scenario is presented in Figure 5.9.

It is seen that the average execution time decreases exponentially if we provide higher-profile VM. Precisely, Medium VM gets approximately 60% less average execution time, while Large VM consumes about 80% less average execution time when compared with Small VM. As stated in Table 5.3, Large VM has four times MIPS as much as Small VM, and Medium VM offers twice MIPS than Small VM. Hence, higher configured VM can provide more computing capacity.

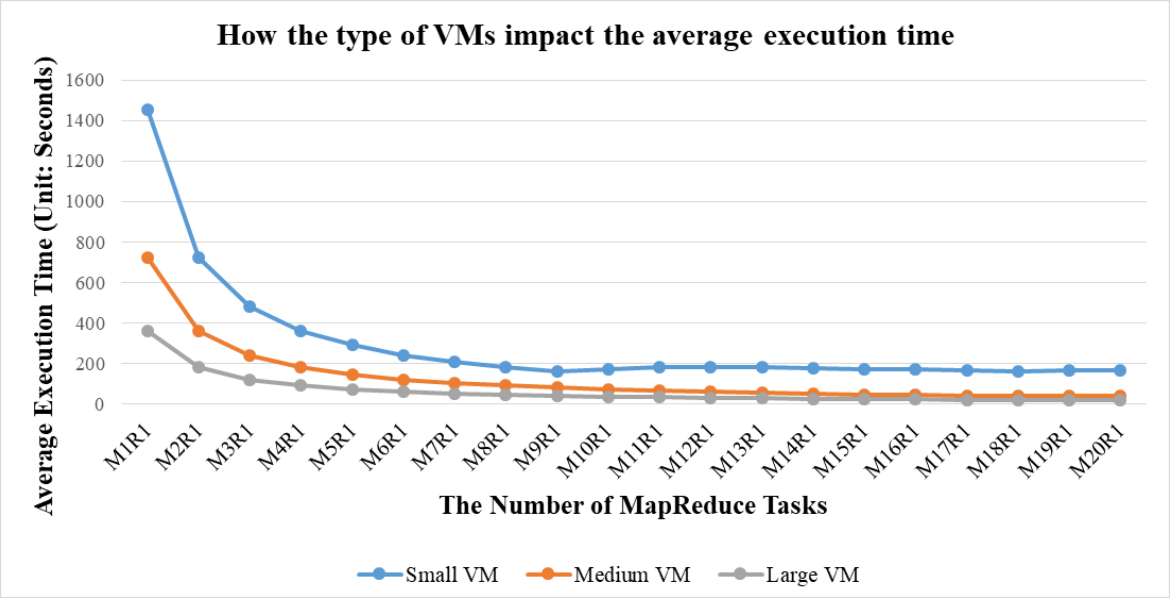


Figure 5.9: How the VM type impacts the average execution time

5.7.4 Variation of the Job Types

In this scenario, VM parameters (e.g., Small VM) and the number of VM (e.g., 3 VMs) are invariant, while job type changes (e.g.,from Small Job to Big Job). The result of this scenario is shown in Figure 5.10.

It is seen from this figure that the total VM costs of processing Big Job is twice as much as that of Medium Job, and the total VM costs of processing Medium Job is twice than that of Small Job. According to Table 5.4, Big Job doubles its job length (MI) than Medium Job while Medium Job double its job length (MI) than Small Job, hence when the same quantity of VMs with identical VM parameters is offered, higher-workload job has linearly increased its total VM cost, which is reasonable in the real world.

In summary, we can conclude from the above observations that the efficacy of IoTSim has been proven through extensive experiments. IoTSim largely extended Cloudsim's functionality to support modeling and simulation of multiple cloud-hosted BDAAs running concurrently in shared cloud datacenters. IoTSim is capable of simulating MapReduce BDAAs to study the correctness and effect of independent variables (e.g.,

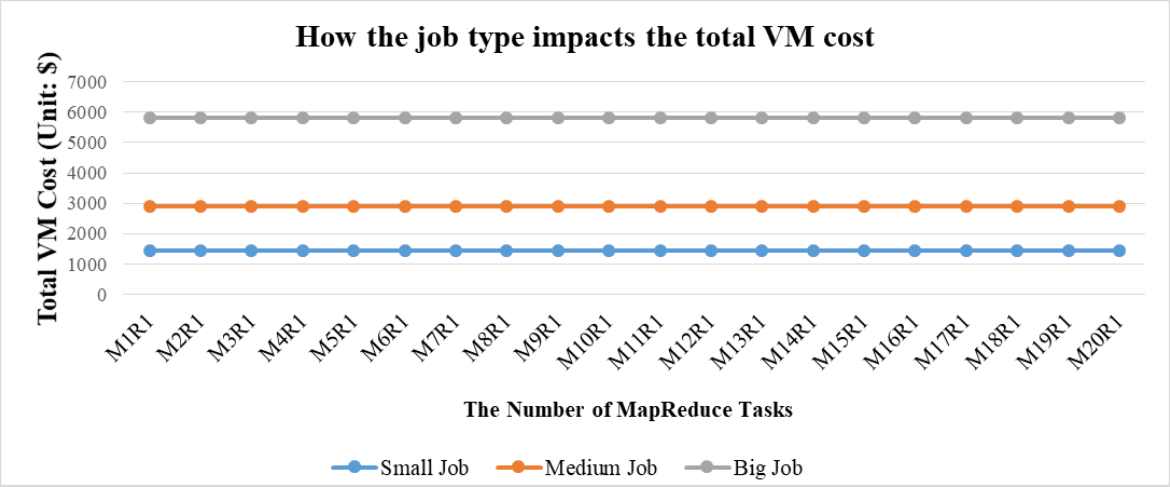


Figure 5.10: How the job type impacts the total VM cost

VM parameters, the number of VM, job parameters, MR combination) on the dependent variables (e.g., average execution time, maximum execution time, minimum execution time, makespan, VM computation cost, and network cost) and achieve a high degree of accuracy. IoTSim enables researchers to analyze how a MapReduce BDAA performs in cloud environment and evaluate if it meets SLAs.

5.8 Summary

In this chapter, we conduct an intensive study on how to use a simulation-based technique to model and simulate cloud-hosted BDAAs. We first analyze the gaps of extant representative simulators. Then, we design and implement a new simulation tool called IoTSim to fill the identified gap. IoTSim has been built on top of a widely used simulator, i.e., Cloudsim, and extensively extend and enhance Cloudsim with many new functions, entities, and events. Users can use IoTSim to model and mimic different cloud features and components such as datacenters, host, VM, storage, and networking. Most importantly, users can successfully model and simulate the features, components, and behaviors of dominant data processing framework such as MapReduce as it does in the real world. Also, IoTSim allows users to perform experiments under non-static

conditions (e.g., availability and workload pattern) in a controllable environment where tests can be re-executed, study the impact and performance of cloud-hosted BDAAs and evaluate if their algorithms or solutions meet SLA requirements.

In the next chapter, we will address the research question defined as *RQ4*.

Chapter 6

SLA-driven Scheduling Big Data Analytics Applications in Public Cloud

In this chapter, we will address the research question defined as *RQ4: How to achieve SLA-driven scheduling for cloud-hosted big data analytics application (BDAAs) across layers while guaranteeing SLA?*

To this end, we designed and implemented a new SLA-driven scheduling algorithm that is capable of achieving SLA-guaranteed schedule for cloud-hosted BDAAs using MapReduce as a processing engine.

6.1 Positioning versus Existing Scheduling Techniques

Although the scheduling-based technique is often used by researchers to address SLA management problem, existing algorithms or solutions using scheduling technique are incapable of handling the increased complexities in BDAAs scenarios in achieving optimal resource allocation, reallocation or scheduling within SLA constraints. Take a typical MapReduce BDAA as an instance, it is extremely tough to resolve dependency between CIaaS-level hardware configurations, deployment plan for BDPaaS-level MapReduce software components and BDSaaS-level application SLA constraints manually. In particular, the hard challenge is to flexibly select CIaaS configurations (I/O capacity, RAM, VM speed, local storage, VM cost, and VM Utilization) for scheduling BDPaaS-level software components (such as the number of Map tasks, the number of Reduce tasks; Map Slots per VM, Reduce Slots for VM, and Max RAM per slot) driven by BDSaaS-level application SLA constraints (e.g., analyzing 100GB of Tweets in 10 minutes while

subjecting to maximum budget of $100). The space of possible configurations for BD processing frameworks and hardware resource is vast so computing an optimal solution is an NP-complete problem, and therefore intractable given current scheduling technology. To be precise, we draw a simple example to demonstrate the challenge as an NP-complete problem: We assume that a cloud-hosted BDAA contains 500 map tasks and 100 reduce tasks, and this application will be deployed in public cloud which has 100 available VMs. Therefore, there are $100^{(500+100)}$ possible allocations for deploying the given application.

Amongst the reviewed papers using scheduling technique, most of the contributions focus on designing time efficient resource provisioning and task scheduling. Lee et al. [344] proposed dynamically allocating public cloud resources to a Hadoop cluster based on a simple SLA constraint: minimize storage size. Kambatla et al. [345] suggested selecting the optimal set of public cloud resources for the Hadoop cluster by developing and profiling hardware resource consumption statistics. Similarly, the authors [346] proposed selecting configurations of heterogeneous Amazon EC2 resources under various what-if scenarios (the number of Map tasks, the number of Reduce tasks, size and distribution of input data). However, none of these works considered deadline and budget SLA constraints while taking scheduling decisions. Mattess et al. [232] proposed a policy for dynamic provisioning of cloud resources to speed up execution of deadline-constrained MapReduce-based BDAA, by enabling concurrent execution of tasks, in order to meet a soft deadline for completion of the map phase of a BDAA. However, they only considered soft deadlines at the map phase while ignoring the reduce phase.

In contrast in this thesis, the execution of map and reduce tasks are scaled across a public cloud, where resources can be virtually unlimitedly, with the goal of meeting user-defined deadline and budget. Moreover, we consider the data transmission time in our model which might significantly affect total makespan due to large data size. Our methodology differentiates itself in the following aspects. First of all, we present a mathematical model that enables holistic modeling of relationship between application-level SLA parameters (e.g., budget and deadline, application size) and MapReduce configurations at BDPaaS level in terms of: (i) data volume (ii) platform

component parameters (the number of mappers, the number of reducers) and CIaaS-level configuration (VM type, VM speed, Memory capacity, VM cost etc.,). Secondly, we develop a greedy heuristic based MapReduce application scheduling algorithm that can pro-actively minimize the cost under user's SLA constraints (budget and deadline), data workload and processing and performance (e.g., data center availability, throughput, and VM utilization) uncertainties. We compare its performance with exhaustive search algorithm and NoSLA approaches.

To best of our knowledge, it is the very earliest work that models all the requirements of a MapReduce-based BDAAs (both computing and communication) and schedules them in public cloud to minimize their execution cost while respecting SLA objectives such as budget and deadline.

6.2 Cross-layer System Model for SLA-driven Scheduling BDAAs in Public Cloud

We first introduce a high-level system model for SLA-driven scheduling BDAAs in public cloud, which consists of scheduling mechanism along with its actors in different layers shown in Figure 6.1. End users on submitting their SLA requirements request the service from a BDSaaS provider. In response to the incoming requests, a BDPaaS provider analyzes this specific requirement of SLA parameters and decide upon what is the optimal strategy regarding fulfillment of the request based on the potentiality, availability, and price of VMs that he rents from CIaaS provider. Then, the scheduling component is responsible for allotting resources that are based on the above "smart" decision. The following section gives the details of the system model and scheduling scenario.

BDSaaS layer: Various applications are encapsulated in the form of service and offered to end users. Such users submit requests to a BDSaaS provider. The users' request is featured by application workload with specific SLA objectives such as deadline by which his/her MapReduce jobs must be finished and a budget which he/she is willing to spend. SLA requirements include the following properties:

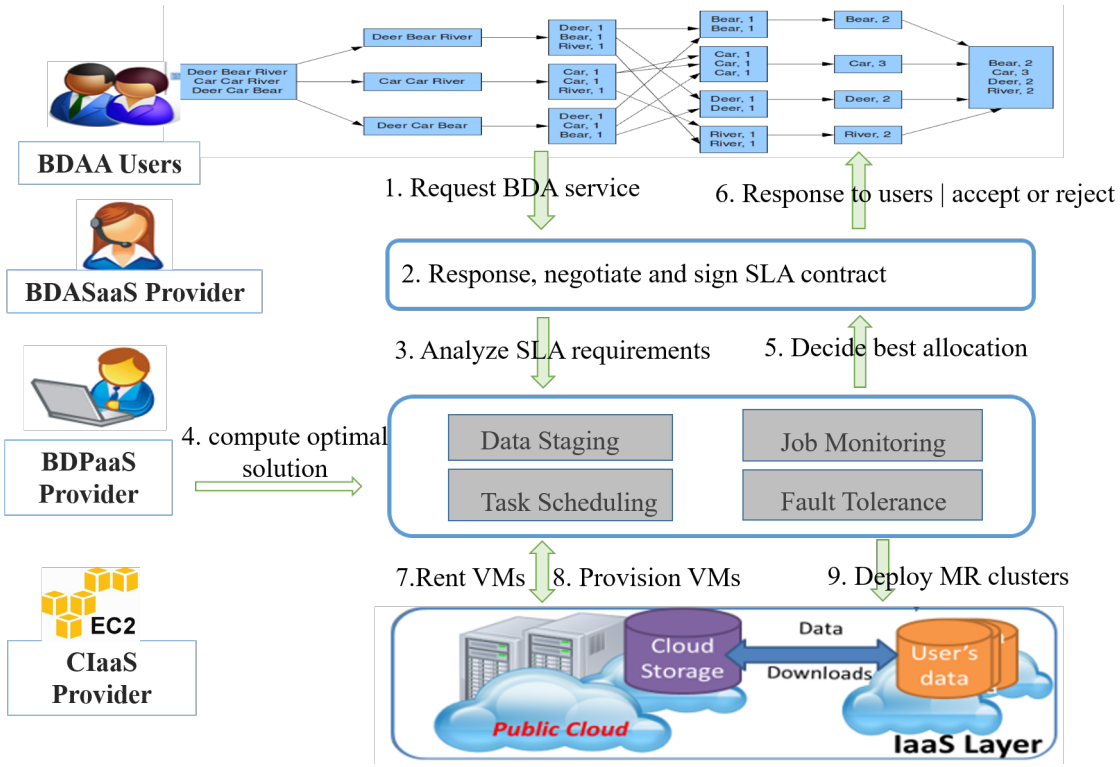


Figure 6.1: A high-level system model for scheduling BDAAs in public cloud

- Deadline: Maximal time user would like to wait for the result.
- Budget: The amount that a user is willing to pay for the requested services
- Penalty Rate: The amount given for customers' compensation when provider misses the deadline
- Input File Size: The size of users' input file
- Request Length: The amount of Millions of Instructions (MI) are required to be executed to serve the particular user's request

BDPaaS layer: A BDPaaS provider hires appropriate datacenter resources from CIaaS provider for deploying the platform to support the application running, while meeting agreed on SLA objectives. The BDPaaS provider is responsible for analyzing users' requests and scheduling platform-level resources according to aSLA constraints. BDPaaS provider aims at minimizing functional cost by efficiently using resources from

CIaaS providers and improving customer satisfaction level by providing parameters of SLAs that are used to guarantee the quality of service requirements of accepted users. In order to guarantee SLAs, the BDPaaS provider has to optimally and strategically use the platform resources by leveraging our proposed scheduling algorithm. Specifically, the BDPaaS provider will decide which type of VMs to be utilized so that the cost of execution can be minimized. It will also decide where (e.g., type of VM, which VM) each map and reduce task should be placed and executed. After analyzing the user's request, the BDPaaS provider deploys a MapReduce cluster after negotiating all the required cloud infrastructure from the CIaaS provider. SLA requirements include the following properties:

- The number of map task
- The number of reduce task
- The number of instances if applicable
- Blocksize
- The input data size for map task
- The input data size for reduce task

CIaaS layer: Public cloud provider such as Amazon offers VMs to BDPaaS provider, and it hides low-level details of dispatching images of VMs to run on their physical resources. SLA requirements include:

- Service Initiation Time: Time taken to deploy a VM
- Cost: The amount the BDPaaS provider has to pay per hour for using a VM from a CIaaS provider
- Input Data Transfer Cost: The amount the BDPaaS provider has to pay for data transfer from users' local machine to CIaaS provider's VM
- Output Data Transfer Price: The amount the BDPaaS provider has to pay for data transfer from CIaaS provider's VM to user's local machine

- Processing Speed: The speed at which VM is processing. MIPS is used as a unit of a VM's processing speed. The processing speed varies according to different VM type (i.e., small VM, Medium VM and large VM)

- Data Transfer Speed: The speed at which the data is transferred. It relies on the location distance and also the network performance

In the next section, we will discuss the mathematical model and assumption that is considered as part of our scheduling approach.

6.3 Mathematical Model and Assumptions on the Monetary Cost

For clarity and quick reference, we develop a set of mathematical symbols to characterize elements in our model frequently used hereafter in Table 6.1.

6.3.1 MapReduce Job Model

We model that a MapReduce Job J_i that consists of a set of map tasks M_i and a set of reduce tasks R_i, where $|M_i| \geq |R_i|$. The aggregate data size $\in J_i$ is be represented as $Size(J_i)$. It is also the total size of input datasets of the map tasks, hence $Size(J_i) \equiv Size_{map}(J_i)$. On the other hand, the total size of the input data of the reduce tasks is denoted by function $Size_{reduce}(J_i)$.

Furthermore, in this thesis we consider three types of VMs which can host map or reduce tasks: $VM = [Small\,VM, Medium\,VM, Large\,VM]$, and the MIPS (millions of instructions per seconds) rating of these VMs is denoted by set VM_{mips}. We assume that a $Small\,VM$ can only run one map task or one reduce task at a given point of time, i.e., $Small\,VM_{map} = 1$ and $Small\,VM_{reduce} = 1$. Proportionally, $Medium\,VM_{map} = 2$ and $Medium\,VM_{reduce} = 2, Large\,VM_{map} = 4$ and $Large\,VM_{reduce} = 4$ respectively.

Our modeling assumptions are based on Amazon EC2 [92] VM configurations where the number of processor core doubles across VM types. It is a reasonable and practical assumption since today many users rent public cloud resources (Amazon EC2 is the most popular public cloud as it is observed from systematic literature review work in

Table 6.1: Notation used in the mathematical model of the monetary cost

Symbols	Description
MapReduce Workload	
J	a set of MapReduce jobs or workloads
J_i	a MapReduce job or workload instance in J. For generality, each J_i has different user-specified attributes (e.g., M_i, R_i etc.,)
M_i	a set of map tasks $\in J_i$
R_i	a set of reduce tasks $\in J_i$
$Size_{map}(J_i)$	the input data size of map tasks $\in J_i$
$Size_{reduce}(J_i)$	the input data size of reduce tasks $\in J_i$
$MI_{map}(J_i)$	the millions of instructions per MB data when processing each map task
$MI_{reduce}(J_i)$	the millions of instructions per MB data when processing each reduce task
$Blocksize$	the default data block size that a distributed file system can store
D	a set of deadline for J
B	a set of budget for J
D_i	the deadline of J
B_i	the budget of J
VM Parameters	
VM	a set of VM types
VM_{mips}	set denoting MIPS rating of VMs
VM_{map}	the upper limit on the number of map tasks that can be mapped to VM
VM_{reduce}	the upper limit on the number of reduce tasks that can be mapped to VM
$VM_{bandwidth}$	VM's network bandwidth
Y	the leasing cost of a small VM type for an hour
VM_j	a VM instance j
N_i	the number of VM that has been allocated for J_i
Makespan	
$TT(M_i, VM_j)$	network delay (transfer cost) in transferring input data of map tasks of J_i to VM_j
$TT(R_i, VM_j)$	network delay (transfer cost) in transferring input data of reduce tasks of J_i to VM_j
VM_j^{bw}	the network bandwidth of VM_j
$MIMT(M_i)$	J_i's aggregated millions of instructions(MI) of map tasks
$TET(M_i, VM_j)$	the execution time of the map tasks of J_i
$MIRT(R_i)$	J_i's aggregated millions of instructions(MI) of reduce tasks
$TET(R_i, VM_j)$	the execution time of the reduce tasks of J_i
$Makespan(J_i, hVM)$	the total makespan of executing job J_i over hVM VMs
$Makespan(J_i, hVM^{'})$	the makespan of executing map tasks M_i of job J_i over $hVM^{'}$ VMs
$Makespan(J_i, hVM^{''})$	the makespan of executing map tasks M_i of job J_i over $hVM^{''}$ VMs
Monetary Cost	
$Cost(J_i, hVM)$	the cost of execution J_i over hVM VMs

Chapter 3) instead of maintaining their own private resources to run the applications. Our modeling assumptions are based on Amazon EC2 VM configurations where the number of processor core doubles across VM types. It is a reasonable and practical assumption since today many users rent public cloud resources (Amazon EC2 is the most popular public cloud as it is observed from systematic literature review work in Chapter 3) instead of maintaining their own private resources to run the applications.

We also assume that each VM can be allocated network bandwidth in proportion to their sizes. For example, the bandwidth allocation of each VM type is defined by the relation: $4Small\,VM_{bandwidth} = 2Medium\,VM_{bandwidth} = Large\,VM_{bandwidth} = 4B(Mbps)$, where $VM_{bandwidth} = [Small\,VM_{bandwidth}, Medium\,VM_{bandwidth}, Large\,VM_{bandwidth}]$.

In the following subsections, we first model the makespan of MapReduce job followed by the monetary cost model. Finally, we formalize the multiple-objective optimization problem.

6.3.2 Makespan of MapReduce Job

As we discussed above, each type of VM has its corresponding capacity limit for processing map and reduce tasks shown in Eq. 6.1 and Eq. 6.2 respectively:

$$VM_{map} = [Small\,VM_{map}, Medium\,VM_{map}, Large\,VM_{map}] \tag{6.1}$$

$$VM_{reduce} = [Small\,VM_{reduce}, Medium\,VM_{reduce}, Large\,VM_{reduce}] \tag{6.2}$$

In general, MapReduce jobs are processed in four stages [101]. In the first stage, input data is transferred to a MapReduce cluster. In the second stage, Map tasks process the data. In the third stage, the shuffling of intermediate data is done and in the last stage Reduce tasks aggregate the result set emitted by different Map tasks. Based on these four stages, we can model the makespan of a MapReduce job by splitting it into four steps: map data transfer, map task execution, reduce data transfer and reduces task execution. Figure 6.2 denotes the composition of makespan by taking an example of provisioning three VM resources (which includes one $Large\,VM$, $Medium\,VM$, and $Small\,VM$ respectively) and running a MapReduce job with three map tasks and two reduce tasks.

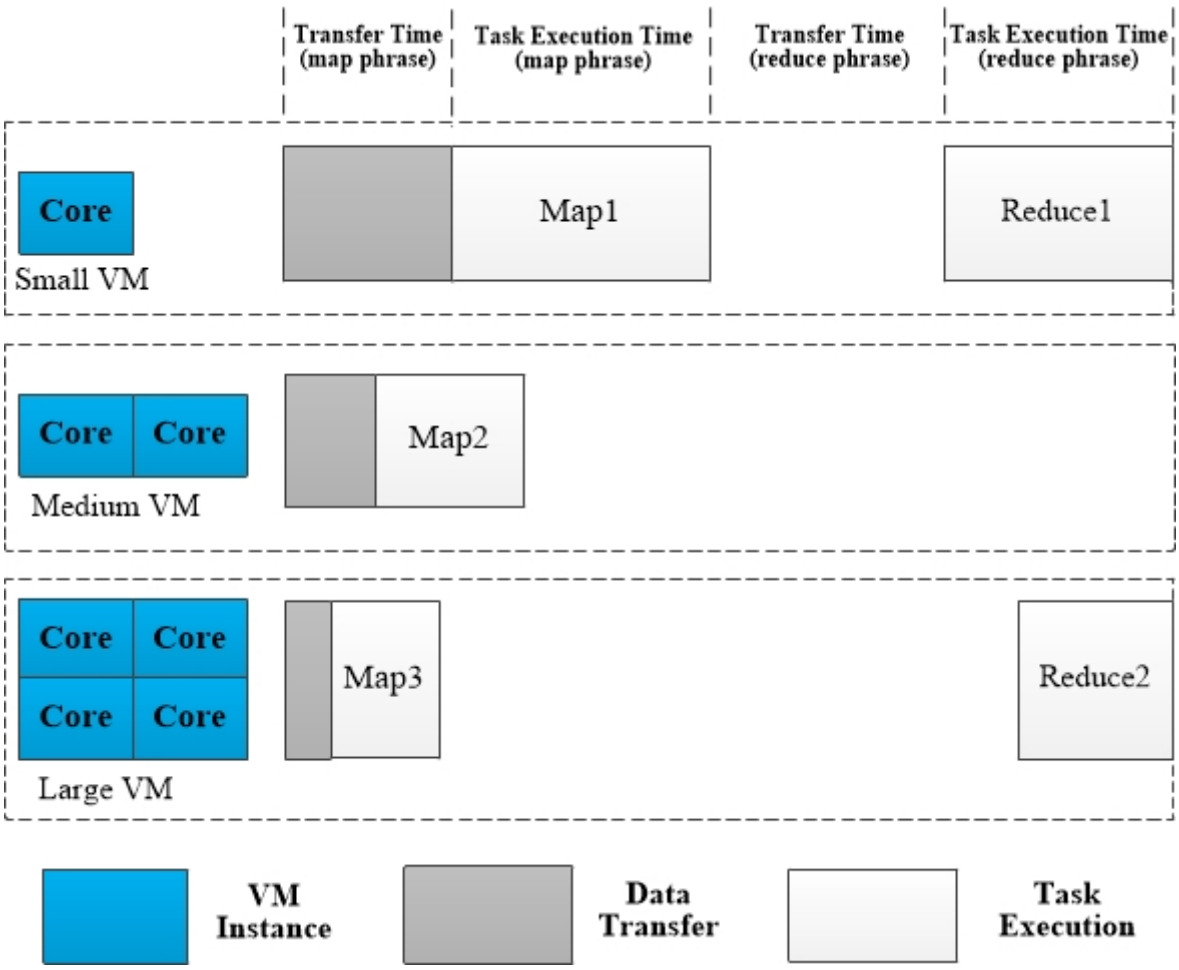


Figure 6.2: The compositions of makespan when executing a MapReduce job J

6.3.2.1 The Network Delay of Transferring Input Data to Map Task

Given a MapReduce job J_i, we apply the Eq. 6.3 to calculate the network data transfer delay:

$$TT(M_i, VM_j) = C_0 + C_1 * \frac{\frac{Size_{map}(J_i)}{N_i}}{VM_j^{bw}}, \qquad VM_j^{bw} \in VM_{bandwidth} \tag{6.3}$$

, where C_0 and C_1 are constants.

6.3.2.2 The Execution Time of a Map Task

In order to compute the execution time of map tasks, we model *Millions of Instructions of Map Task (MIMT)* in Eq. 6.4:

$$MIMT(M_i) = MI_{map} * \frac{Size_{map}(J_i)}{|M_i|} \tag{6.4}$$

As denoted in Table 6.1, MI_{map} is the millions of instructions per MB data when processing each map task. Therefore, the execution time of a map task on a given VM_j is:

$$TET(M_i, VM_j) = \frac{MIMT(M_i)}{VM_j^{mips}}, \qquad VM_j^{mips} \in VM_{mips} \tag{6.5}$$

6.3.2.3 The Network Delay of Transferring Input Data to Reduce Task

According to MapReduce working mechanism, the number of reduce tasks of J_i is always less than the number of map tasks. Moreover, the data size of the reduce tasks must also differ from that of map tasks. Thus, the network delay of transferring data to VM_j where a reduce task will be executed is:

$$TT(R_i, VM_j) = C_2 * N_i + C_3 * \frac{Size_{reduce}(J_i)}{VM_j^{bw}}, \qquad VM_j^{bw} \in VM_{bandwidth} \tag{6.6}$$

, where C_2 and C_3 are constants.

6.3.2.4 The Execution Time of a Reduce Task

Similar to map task, we calculate the execution time of a reduce tasks on a VM using the concept: *MIRT (millions of instructions of reduce task)*, which is defined as:

$$MIRT(M_i) = MI_{reduce} * \frac{Size_{reduce}(J_i)}{|R_i|} \tag{6.7}$$

As denoted in Table 6.1, MI_{reduce} is the millions of instructions per MB data when processing each reduce task. Therefore, the execution time of a reduce task on a given VM_j is:

$$TET(R_i, VM_j) = \frac{MIRT(R_i)}{VM_j^{mips}}, \qquad VM_j^{mips} \in VM_{mips} \tag{6.8}$$

A MapReduce Job J_i may not be executed in a single VM, the makespan of J_i over a set of map and reduce tasks mapped to VMs hVM can be computed as shown in Eq. 6.9. The hVM has two subsets $hVM^{'}$ and $hVM^{''}$, representing those VMs that execute map and reduce tasks respectively.

$$Makespan(J_i, hVM) = Makespan_{map}(M_i, hVM^{'}) + Makespan_{reduce}(R_i, hVM^{''})$$
$$= \max_{VM_j \in hVM^{'}}[TT(M_i, VM_j) + TET(M_i, VM_j)] + \max_{VM_h \in hVM^{''}}[TT(R_i, VM_h) + TET(R_i, VM_h)] \tag{6.9}$$

, where $|M_i| = \sum_{VM_j \in hVM'} VM_j^{map}$ and $|R_i| = \sum_{VM_h \in hVM''} VM_h^{reduce}$.

Here, VM_j^{map} is the maximum number of map tasks that VM_j can host, and VM_h^{reduce} is the maximum number of reduce tasks that VM_h can host.

6.3.2.5 Monetary Cost

In this thesis, we have applied the price schema of Amazon EC2 [92] to estimate the cost per hour of using a hosted VM. We assume that VMs are charged under a pay-as-you-go model (e.g., per minute). For example, the price of hiring or leasing the computation time of a $Small\,VM$ for 3 hours at Y dollar per hour base rate will be $3 * Y$ dollars. Accordingly, in our model, the $Medium\,VM$ and $Large\,VM$ cost $2 * Y$ dollars and $4 * Y$ dollars per hour respectively.

In summary, the total cost of processing a MapReduce job J_i on the set of heterogeneous VMs will be:

$$Cost(J_i, hVM) =$$
$$Makespan_{map}(M_i, hVM^{'}) *$$
$$\sum_{VM_j \in hVM'} (Y * [VM_j \in Small\,VM] + 2Y * [VM_j \in Medium\,VM] + 4Y * [VM_j \in Large\,VM])$$
$$+ Makespan_{reduce}(R_i, hVM^{''}) *$$
$$\sum_{VM_h \in hVM''} (Y * [VM_h \in Small\,VM] + 2Y * [VM_h \in Medium\,VM] + 4Y * [VM_h \in Large\,VM] \tag{6.10}$$

6.3.3 Multi-objective Optimization Problems

It is evident from Eq. 6.4 and Eq. 6.7 that by increasing the level of parallelism for map and reduce tasks (i.e., hiring more and more number of VMs), the overall makespan can be reduced. Furthermore, using more powerful VMs (Large vs. Small) has further potential to improve the makespan due to superior processor speed and network I/O capacity. However, adding more VMs and replacing small VM with larger VM will certainly lead to elevated monetary cost. In other words, there exists a trade-off between makespan and monetary cost for a MapReduce-based BDAA given the budget and deadline constraints.

In this thesis, we consider the following optimization problem: How to minimize the monetary cost to process a given set of MapReduce jobs J while meeting the deadline D and budget B. The problem is formalized as:

$$\arg\min \sum_{J_i \in J} COST(J_i, hVM)$$

$$Subject\ to:$$

$$Makespan(J_i, hVM) \leq D_i\ and\ Cost(J_i, hVM) \leq B_i$$

$$, where\ \mathrm{D}_i \in D\ and\ B_i \in B \tag{6.11}$$

6.4 Proposed MapReduce Application Scheduling Algorithm

The provisioning and scheduling problem discussed above is a multidimensional knapsack problem that was shown to be NP-complete as they map to 0-1 Knapsack problems [347]. Thus a heuristic approach is necessary to solve the problem. In this section, we present the details of MapReduce application scheduling algorithm (MASA) with the goals of optimizing Eq. 6.11 considering the budget and deadline SLA constraints.

In MapReduce frameworks such as Hadoop, in general, data is distributed across several cluster nodes where map tasks are scheduled in a round-robin fashion in order to have balanced load across each cluster node. In other words, each node will execute a more or less equal number of map tasks considering homogeneous configuration across

nodes. Thus, without loss of generality, we can assume there is one large size map task running on each node instead of several small map tasks.

Algorithm 1 describes the significant steps of MASA algorithm. In the first step, the lower bound and upper bound number of map tasks are calculated such that user-specified deadline and budget constraints can be achieved. For the sake of simplicity, the lower bound of map tasks equals one. The upper bound of map tasks can be calculated by $\frac{Size_{map}(J_i)}{Blocksize}$. $Size_{map}(Ji)$ means the input data size of map tasks J_i. Blocksize represents the default data block size that a distributed file system can store. The second step is to decide the scope of reduce task number. Since the number of reduce task is smaller than the number of a map task, we set the lower bound of reduce tasks as one and upper bound of reduce tasks as the number of map task. The main step followed is to compute allocation based on the pair of map tasks and reduce tasks for the subsequent requests. Specifically, MASA iteratively computes (in a greedy fashion) all the possible number of map tasks and reduce tasks according to the above lower bound and upper bound. Then, a number of same types VMs ($SmallVM$ or $MediumVM$ or $LargeVM$) are calculated. After that, MASA allocates each possible pair of map and reduce tasks into these VMs. For each potential allocation, makespan and cost are calculated according to the formulas discussed in the above subsections. In each iteration, MASA discards the "bad" allocation that has either makespan exceeding deadline or cost going beyond budget. After the iteration, a set of valid allocations is collected and stored in $AllocatedList$. Finally, we pick the allocation which has the minimal monetary cost from $AllocatedList$ and deploys the jobs based on this optimal allocation $Allocation^{optimum}$.

Time complexity: we assume that N^{map} number of map jobs meets the lower and upper bound, and M number of available VMs, so the time complexity for allocating the map jobs in the worst case is $O(N^{map} * M)$. Similarly, if the number of reduce jobs is N^{reduce}, therefore allocating the given reduce jobs in the worst case the time complexity is $N^{reduce} * M$.

Algorithm 1: MapReduce Application Scheduling Algorithm (MASA)

Input: User Request = r; // user-specified MapReduce-based BDAA and SLA requirements details (i.e., deadline and budget)
Output: optimum solution(s) $Allocation^{optimum}$ allocating map and reduce tasks to VMs in datacenters.

1: Min_{map} = calculate lower bound on the number of mappers
2: Max_{map} = calculate upper bound on the number of mappers
3: Let *AllocatedList* be the list of all possible allocations whose makespan and cost meet SLA requirements
4: **for** i=$Min_{map} : Max_{map}$ **do**
5: **for** *j=1:i* **do**
// for allocation, calculate the number for the same type of VM;
6: a_{ij} = search_allocations(i, j, r);
// calculate makespan and cost of the allocation from a_{ij}
7: calculate makespan(r, a_{ij});
8: calculate cost(r, a_{ij});
// compare the makespan with the deadline of this request r
9: **if** ($makespan \leq deadline$ && $cost \leq budget$) **then**
10: save this candicate allocation into *AllocatedList*;
11: **end**
12: **end**
// choose the allocations $Allocation^{optimum}$ whose cost are minimum among *AllocatedList*
13: choose mimimum allocations(*AllocatedList*);
14: deploy user request r based on the optimum allocations $Allocation^{optimum}$;
15: **end**

6.5 Evaluation

In this section, we evaluate the performance of our proposed MASA algorithm. For better demonstration, we use two algorithms as benchmarks. The first one exists MapReduce scheduling approach (NoSLA or SLA agnostic). We refer to the existing approach as NoSLA algorithm. The second one is the exhaustive search algorithm which gives a solution close to optimal. We comprehensively compare MASA against NoSLA algorithm and exhaustive search algorithm in the following sections.

6.5.1 Benchmarking Algorithms

6.5.1.1 NoSLA Algorithm: SLA Agnostic Approach

NoSLA algorithm (see Algorithm 2) describes an agnostic MapReduce scheduling approach without considering SLAs. In this algorithm, rather than specifying budget and deadline, a user will randomly specify which types of VM ($SmallVM, MediumVM$ or $LargeVM$) he/she intends to initiate to run his/her MapReduce-based BDAA. The BDPaaS provider will calculate the maximum number of VMs required ensuring the MapReduce job finished before the deadline specified by users. Then they schedule these jobs into the initiated VMs in a simple way. As a consequence, NoSLA algorithm may meet the deadline, but the cost goes very high, or the cost may fall within the scope of the budget, but the deadline cannot be met. Apparently, NoSLA algorithm does not target the trade-off between budget and deadline, which is not able to find the optimal solution from both budget and deadline constraints.

Algorithm 2: NoSLA Algorithm

Input: User Request = r; // user-specified MapReduce-based BDAA and SLA requirements details (i.e., deadline and budget)

Output: allocation a_{ij}; // possible solution(s) allocating map and reduce tasks to VMs

1: $NumberOfVMs = \frac{budget}{VMType^{cost} * deadline}$ // naively calculating the quantity of VMs that need to be initiated

2: a_{ij} = compute_allocation($NumberOfVMs, r$); a_{ij} saves all possible allocations($NumberOfVMs, r$) without considering SLA

3: deploy user request r based on a_{ij};

6.5.1.2 Exhaustive Search Algorithm

Algorithm 3 denotes the procedures how exhaustive search algorithm works. First, the lower bound and upper bound number of map tasks are calculated such that user-specified deadline and budget constraints can be achieved. Following, in each iteration, the exhaustive search algorithm selects all possible VM set that consists of the aforementioned different type of VMs ($SmallVM, MediumVM$ and $LargeVM$). In the main step of computing allocation, the algorithm seeks the best plausible VMs set upon all the possible pair of map tasks quantity and reduces tasks quantity that can not only

Algorithm 3: Exhaustive Search Algorithm

Input: User Request = r; // user-specified MapReduce-based BDAA and SLA requirements details (i.e., deadline and budget)
Output: optimum solution(s) $Allocation^{optimum}$ allocating map and reduce tasks to VMs in datacenters.

```
1:  Min_map = calculate lower bound on the number of mappers
2:  Max_map = calculate upper bound on the number of mappers
3:  Let AllocatedList be the list of all possible allocations whose makespan and cost meet SLA
      requirements
4:  for i=Min_map : Max_map do
5:      for j=1:i do
            // For allocation, select possible VM set = [SmallVM, MediumVM, LargeVM]
               where its' total capacity of processing the map tasks equals i
6:          select_VMset(SmallVM, MediumVM, LargeVM);
7:          a_ij = search_allocations(i, j, r);
            // calculate makespan and cost of the allocation from a_ij
8:          calculate makespan(r, a_ij);
9:          calculate cost(r, a_ij);
            // compare the makespan with the deadline of this request r
10:         if (makespan ≤ deadline && cost ≤ budget) then
11:             save this candicate allocation into AllocatedList;
12:         end
13:     end
        // choose the allocations Allocation^optimum whose cost are minimum among
           AllocatedList
14:     choose mimimum allocations(AllocatedList);
15:     deploy user request r based on the optimum allocations Allocation^optimum;
16: end
```

minimize the cost but also meet the deadline and budget constraints. The algorithm chooses the optimal allocation by searching all possible allocations, this is why we call it exhaustive search. Theoretically, it will give an optimal solution. However, it will consume more time.

6.5.2 Experimental Setup

To model a real public cloud environment and MapReduce-based BDAA scheduling scenario, we utilized the proposed IoTSim in Chapter 5. Our environment setup considered multiple BDAAs with a different user-specified deadline and budget submitted to BDSaaS provider and executed in the big data processing platform (using MapReduce). The generation of application workload and specific evaluation metrics have been considered in the following subsections.

6.5.2.1 Application Workload Generation

Typically, on the user's side, a request for executing MapReduce-based BDAA consists of details of application workload characteristic along with SLA constraints, such as deadline and budget. Next, we discuss how SLA constraints are modeled in our experiments.

- Deadline is defined as the maximum time (upper bound) that user would like to wait until a job finishes. The deadline is measured in minutes. Deadline is calculated based on the makespan. Let $maxExTime$ and $minExTime$ represent the maximum and minimum estimated makespan of a job. Then, the estimated execution time $\alpha = \frac{maxExTime}{minExTime}$. Based on this, we derived three different classes of the deadline as follows:

$$tight\ deadline = 0.5 * \alpha$$
$$medium\ deadline = \alpha$$
$$relaxed\ deadline = 1.5 * \alpha$$

- The budget represents the money that each user is willing to pay for the processing his/her job. Let $maxCost$ and $minCost$ represent the maximum and minimum cost required to process his/her job. Then, the estimated budget $\beta = \frac{maxCost+minCost}{2}$. Based on this, we derived three different classes of budgets that can be specified by a user as follows:

$$tight\ budget = 0.5 * \beta$$
$$medium\ budget = \beta$$
$$relaxed\ budget = 1.5 * \beta$$

- Data Size is the size of data that will be processed by a job. The unit of data size is MB. For experiments, three types of jobs are considered based on data size: short, medium and long. The medium job has five times more data size than the short job, while the long job has ten times the data size of the short job. Discrete uniform distributions are used to generate the data size per job in workload.

- Size of Map Task is defined by millions of instructions that need to be executed to process each MB (Megabyte) data when processing each map task during the map phase. The unit of this parameter is Millions Instructions (MI) per MB. It is generated using a uniform distribution. It is worth to mention that each job has different MI for map task depending on its characteristics of the job execution during map phrase. Verma et al. [348] extracted a single job profile using an automated profiling tool which can uniquely capture and demonstrate critical performance characteristics of the job execution. It can be deduced that MI for map task represents the nature of the different job.

- Size of Reduce Task is denoted by millions of instructions that need to be executed to process each MB data when processing each reduce task during the reduce phase. The unit of this parameter is MI per MB. It is generated using a uniform distribution. It is worth to mention that each job has different MI to reduce task depending on its characteristics of the job execution during reduce phrase.

6.5.2.2 Evaluation Metrics

As the aim of BDPaaS provider is to reduce the cost without missing the deadline and budget, the following two metrics are quantified during evaluation:

- Average Makespan: The average makespan shows how fast MapReduce jobs are executed. Let MK_i represent the makespan of the job J_i and is calculated using Eq. 6.9. The N is the number of jobs. Then, the average makespan is calculated by the following formulas:

$$Average\ makespan = \sum_{i \in N} \frac{MK_i}{N}$$

- Average Cost: The average cost presents how much does it cost to process MapReduce jobs. Let TC_i represent the cost of processing job J_i and is calculated using Eq. 6.10. Let N is the number of jobs. Then, the average cost is calculated by the following formulas:

$$Average\ cost = \sum_{i \in N} \frac{TC_i}{N}$$

6.5.3 Evaluation Scenarios

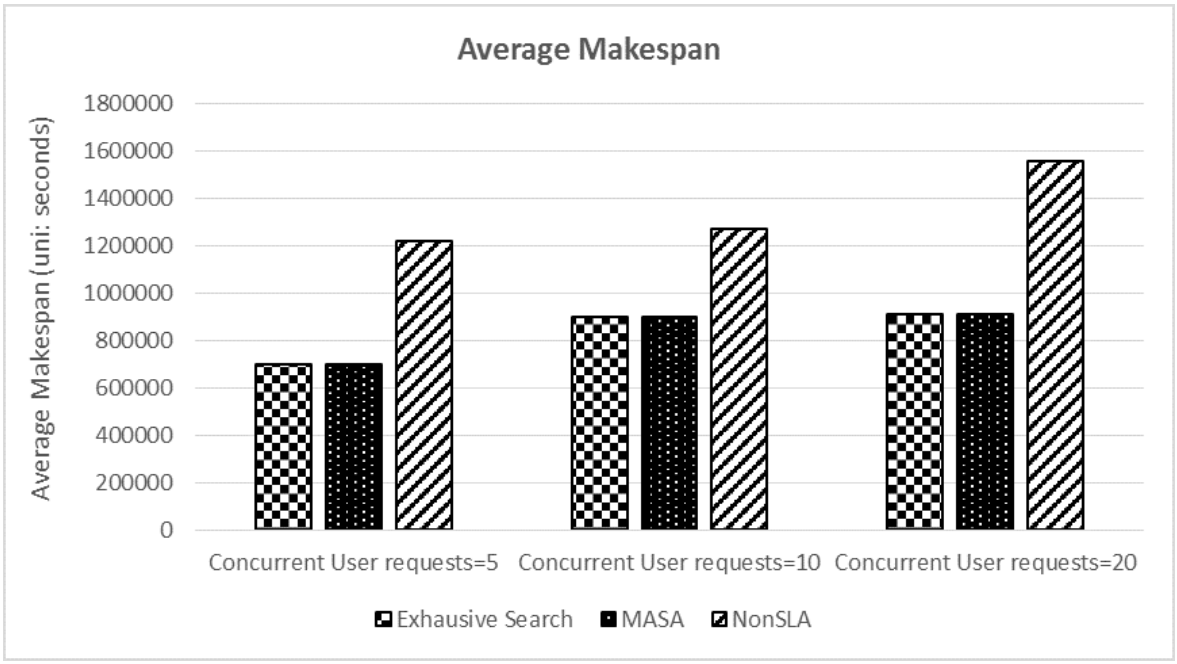
To understand the behavior and performance of our proposed algorithms, we consider the following five different types of scenarios. In each scenario, different experimental parameters discussed above will be applied.

- Variation in the number of concurrent user requests
- Variation in the VM MIPS configurations
- Variation in the Deadlines
- Variation in the Budgets
- Variation in Request Sizes and deadlines

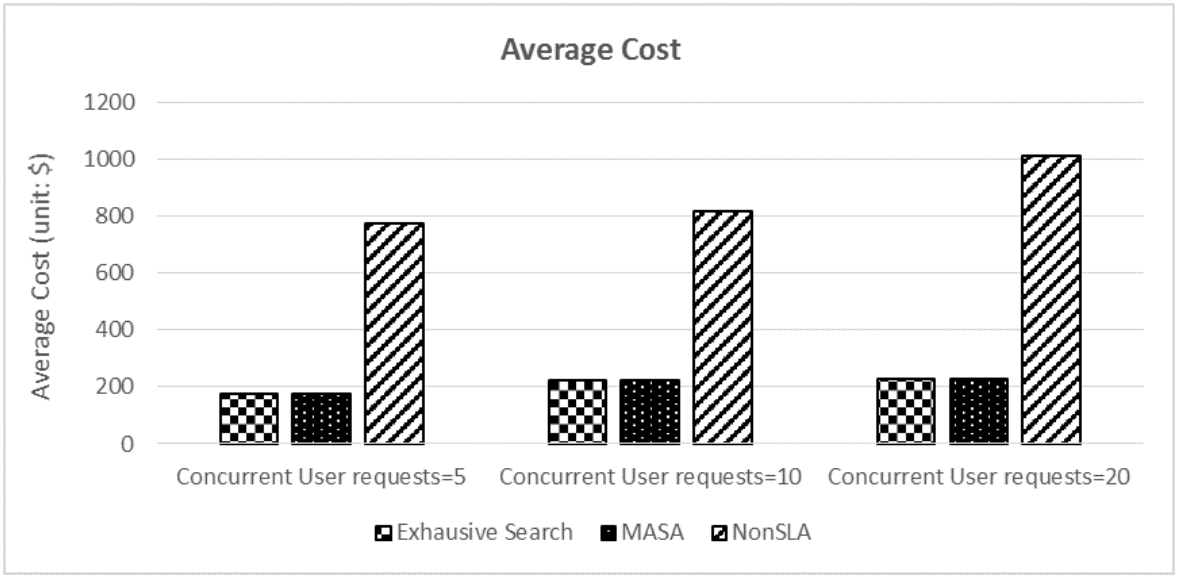
6.6 Analysis of Experimental Results

6.6.1 Variation in the Number of Concurrent User Requests

In this scenario, we vary the number of MapReduce jobs (for example 5, 10 and 20) submitted concurrently while keeping other independent variables (VM parameters, deadline, and budget) at a medium level. Each user request consists of 30% short jobs, 30% medium jobs and 40% long jobs. Figure 6.3 clearly shows how our proposed exhaustive search algorithm and MASA outperform NoSLA algorithm for executing MapReduce jobs in public clouds. MASA shows a very similar performance with exhaustive search algorithm and achieves roughly 25% lower average makespan and 50% average costs in comparison with NoSLA algorithm.



(a) Average Makespan



(b) Average Cost

Figure 6.3: Variation in the number of concurrent user requests (a) Average Makespan; (b) Average Cost

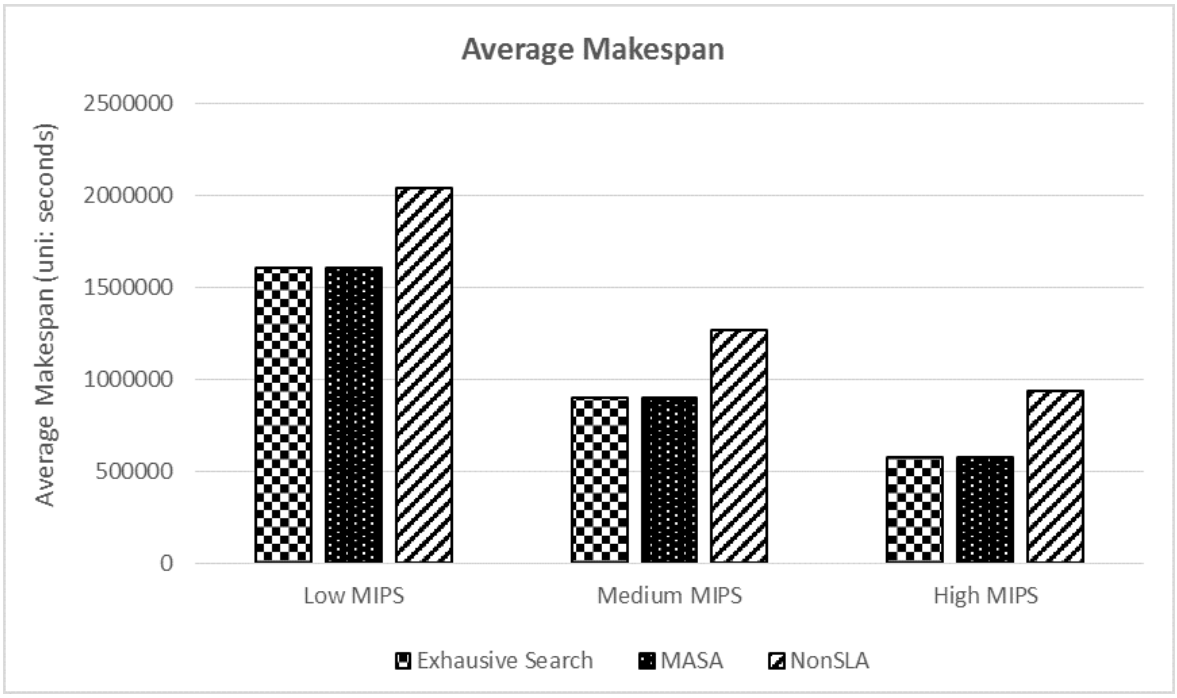
6.6.2 Variation in VM MIPS Rating

In this scenario, we define three different types of MIPS ratings (low, medium, and high). Low MIPS means the MIPS of medium VM is 1.5 times than that of small VM, and the MIPS of large VM is 1.5 times than that of medium VM. Medium MIPS means the MIPS of medium VM is two times that of small VM, and the MIPS of large VM is two times than that of medium VM. High MIPS means the MIPS of medium VM is 2.5 times that of small VM, and the MIPS of large VM is 2.5 times that of medium VM. In experiments, we vary the MIPS rating of VM accordingly while keeping other parameters same.

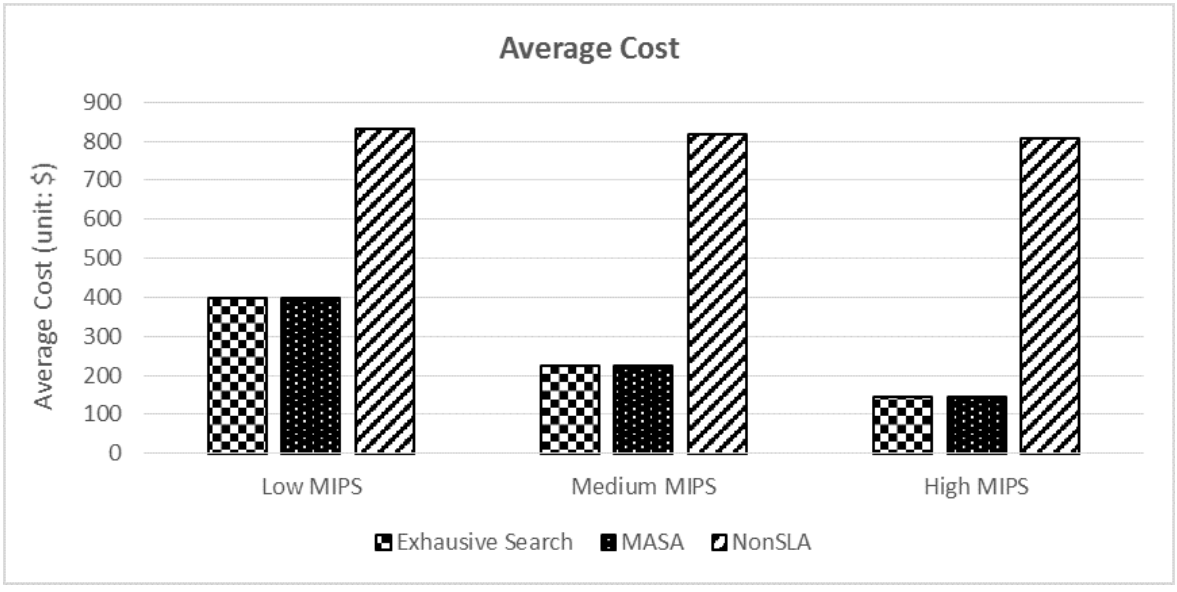
Figure 6.4 shows that overall, MASA still incurs a very low cost to users compared to NoSLA algorithm and very close to the exhaustive search algorithm. Also, it is seen that average makespan decreases considerably as the MIPS rating of VM increases from low end to high end. This is because the disparity of VM's computing power regarding MIPS is expanded and in this case, exhaustive search algorithm and MASA tend to select more large VMs than smaller VMs, causing a considerable decrease of average makespan. However, it is observed that the average cost is appropriately 25% reduced given a high MIPS in comparison with the average cost given a low MIPS. This is because VMs having higher MIPS rating possess higher cost as well, causing the decrease of average cost is less significant than the average makespan when increasing MIPS rating.

6.6.3 Variation in Deadline

In this scenario, we set the aforementioned different types of deadlines (i.e., tight, medium and relaxed) for user requests. Figure 6.5 shows how the performance of MASA, exhaustive search, and NoSLA algorithm is affected by the deadline specified. It is clearly seen that NoSLA approach performance is more than 25% worse than the counterpart of exhaustive search algorithm and MASA regarding average makespan as well as average cost. When the deadline is relaxed, the exhaustive search algorithm achieves just 10% lower average makespan and average cost than MASA. As the deadline becomes stricter, MASA approach performance is similar to the exhaustive algorithm.

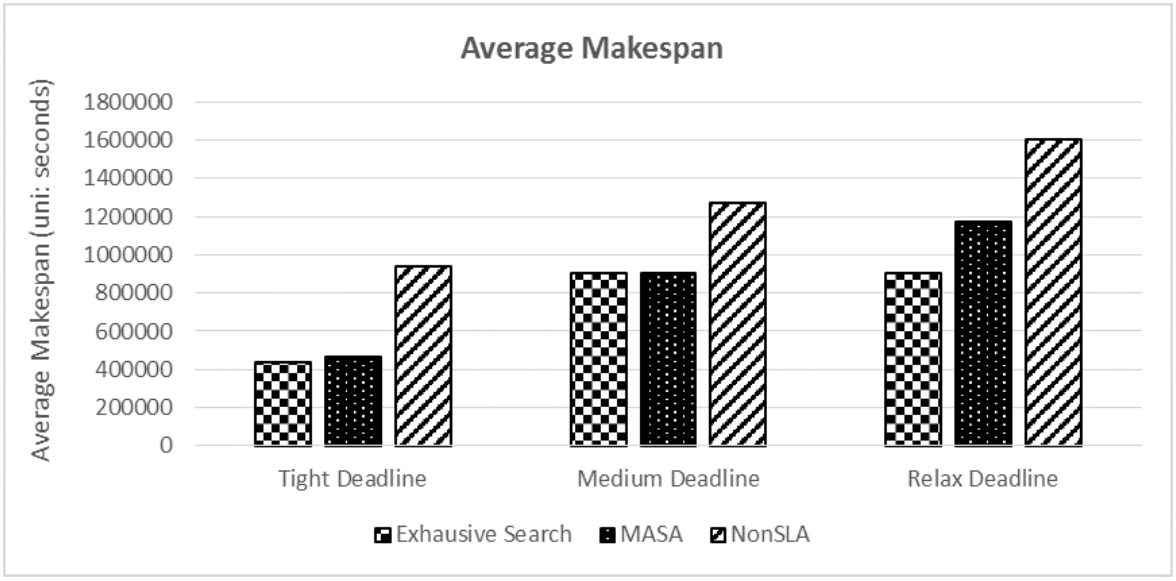


(a) Average Makespan

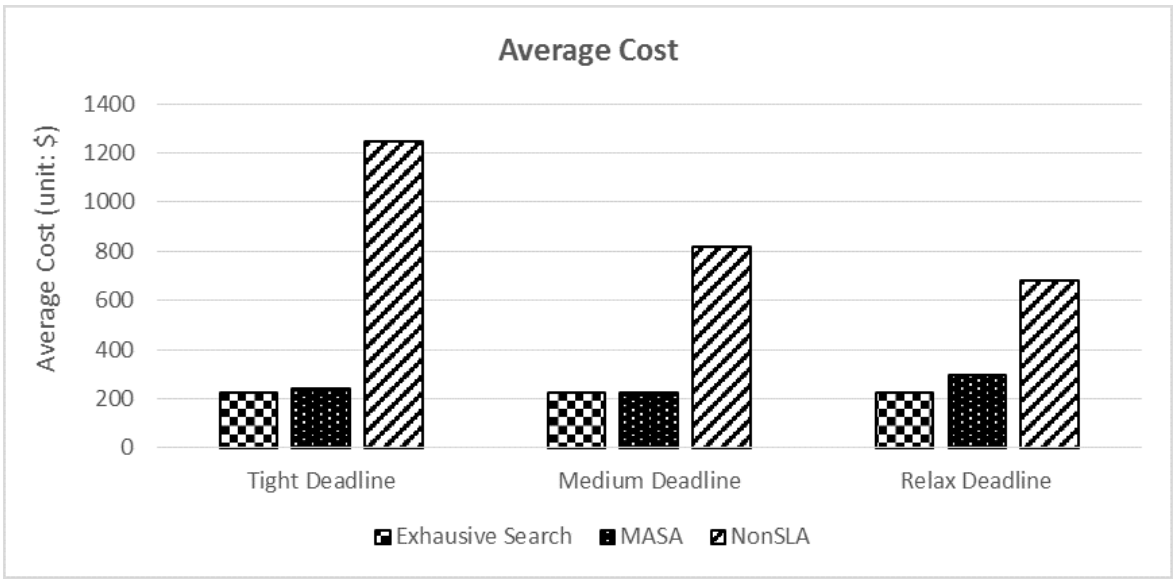


(b) Average Cost

Figure 6.4: Variation in VM MIPS rating (a) Average Makespan; (b) Average Cost



(a) Average Makespan



(b) Average Cost

Figure 6.5: Variation in deadline (a) Average Makespan; (b) Average Cost

The principal reason behind this behavior is that when the deadline specified becomes more tightened, the spaces of possible VMs that host map and reduce tasks are more narrowed and often tilted towards superior VMs types.

6.6.4 Variation in Budget

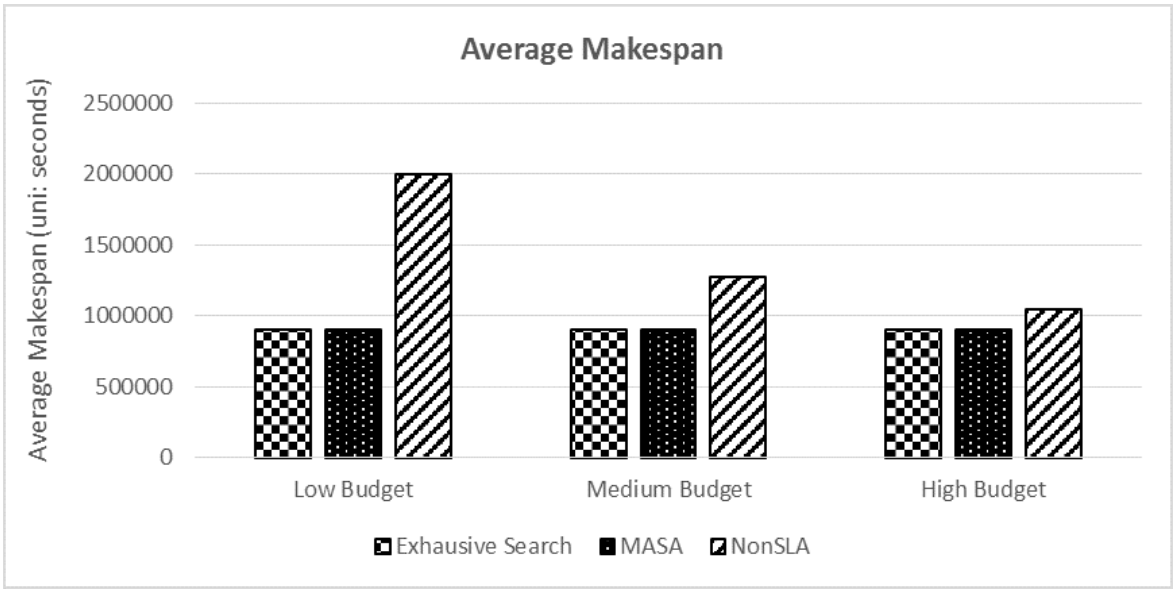
In this scenario, we set the aforementioned different types of budget (i.e., low, medium and high) for user requests. Figure 6.6 shows that NoSLA approach incurs higher average makespan and cost than the counterpart of exhaustive search algorithm and MASA. As the budget is more relaxed, the disparity between NoSLA and MASA regarding average makespan becomes less. This is because when the budget gets relaxed, more VMs that are often tilted towards superior types of VMs are invested to achieve lower average makespan. It is also seen that MASA still achieves an allocation very close to the optimal one.

6.6.5 Variation in Jobs and Deadline

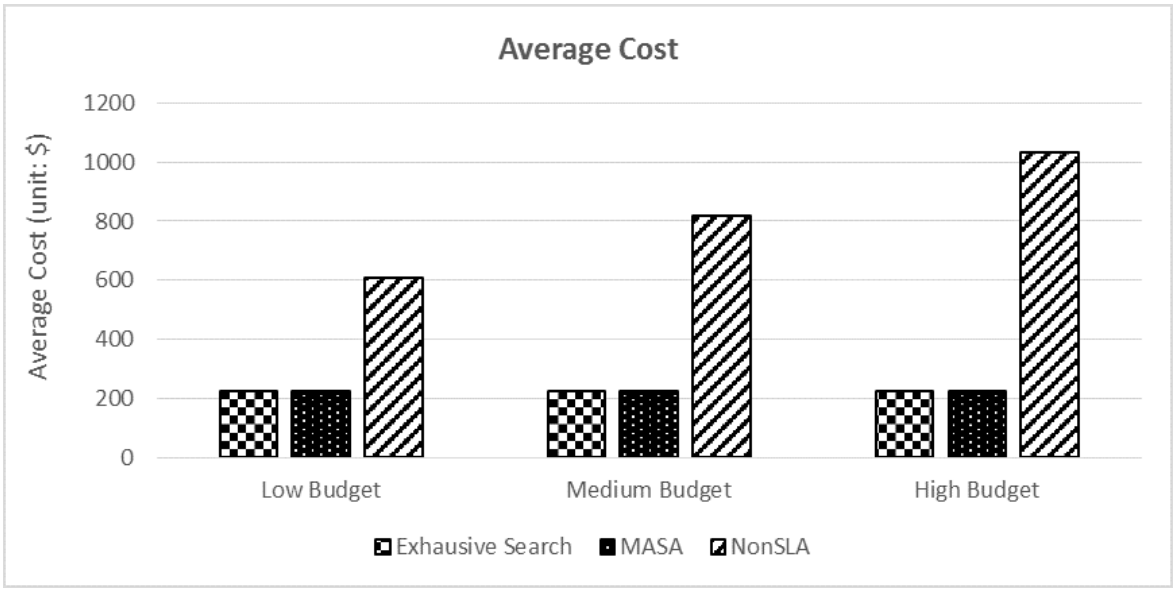
In this scenario, we mix different types of the deadline for different types of jobs while keeping the same VM parameters and cost model. The possible combination of the deadline for different types of jobs is set as follows:

- R4S, M4M, L4T: means we set relax deadline for short jobs, medium deadline for medium jobs and a tight deadline for long jobs
- M4S, T4M, R4L: means we set a medium deadline for short jobs, tight deadline for medium jobs and relax deadline for long jobs
- T4S, R4M, M4L: means we set a tight deadline for short jobs, relax deadline for medium jobs and medium deadline for long jobs

Figure 6.7 shows that still, MASA is better at cost optimization compared to NoSLA algorithm. With a various combination of different deadline types and jobs types, MASA effectively obtains a very close allocation to the exhaustive search algorithm and thus achieves a valid solution that cost within 10% of the optimal performance regarding average cost and average makespan.

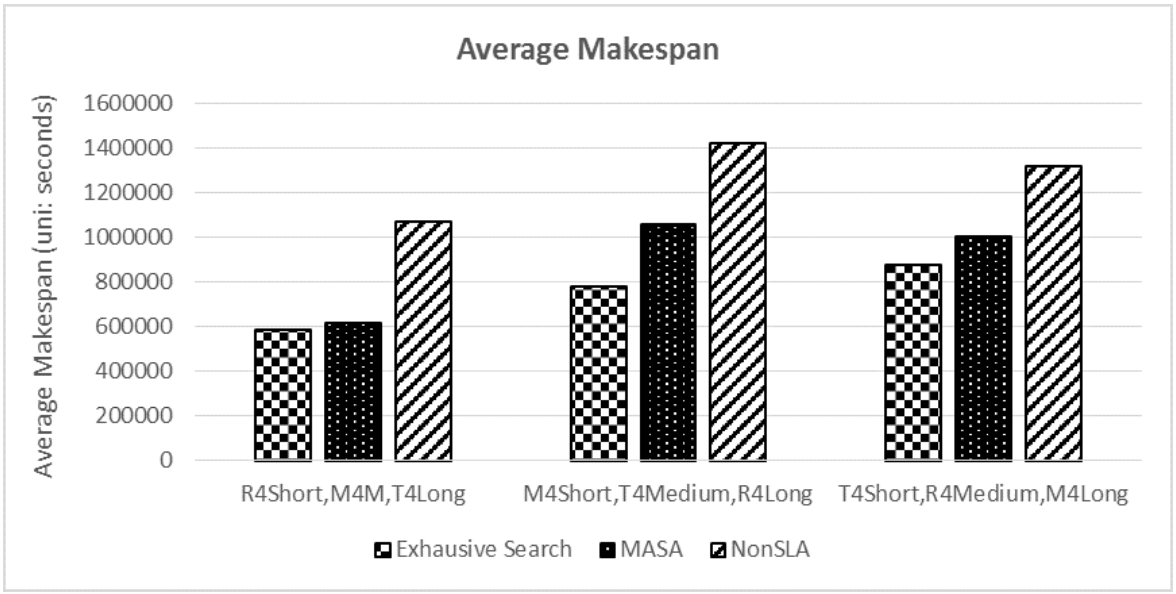


(a) Average Makespan

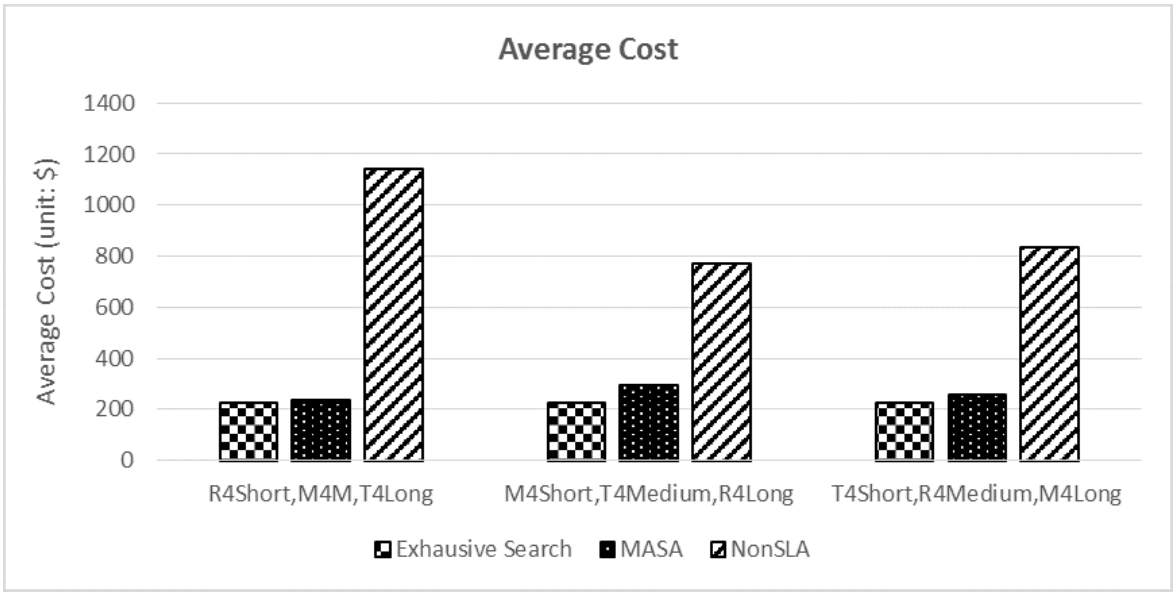


(b) Average Cost

Figure 6.6: Variation in budget (a) Average Makespan; (b) Average Cost



(a) Average Makespan



(b) Average Cost

Figure 6.7: Variation in jobs and deadline (a) Average Makespan; (b) Average Cost

6.7 Summary

This chapter addressed the challenging concerning how to achieve an optimal schedule for MapReduce-based BDAAs in public clouds cross layers while meeting multiple SLA objectives. We first design a mathematical model towards this multi-objective optimization problem and then proposed a new MapReduce application scheduling algorithms which: (i) computes the best combination of different types of VMs greedily; (ii) considers the number of map tasks and number of reduce tasks during the runtime; (iii) deals with run-time uncertainties (e.g., makespan, cost, throughput, and utilization) during resource allocation process and (iv) considers the cross-layer dependency ranging from the CIaaS-level resources to the application-level user requests. Our proposed algorithm successfully minimizes the cost of processing BDAAs while avoiding SLA violations.

The extensive experiments clearly show that MASA can help users reduce the cost of executing the applications in public clouds by about 25% to 50%. The cost-saving efficiency of our proposed SLA-guaranteed scheduling technique depends on the complexity (e.g., the number of map tasks, the number of reduce tasks, input data size, output data size) of the applications. Moreover, it achieves allocation which is very close to the optimal one in most of the scenarios in a more time-efficient way.

In the next chapter, we will address the research question defined as *RQ5*.

Chapter 7

Detection of SLA Violation for Big Data Analytics Applications in Cloud

In this chapter, we will address the research question defined as *RQ5: How to detect SLA violations for cloud-hosted big data analytics applications (BDAAs) across layers before they happen to maximize the providers' profit?*

To this end, we explored multiple machine learning techniques to detect SLA violations for batch workload BDAAs in cloud, which allows us to uncover the hidden pattern of multiple configurations across layers.

7.1 Positioning versus Existing SLA Violation Detection Techniques

SLAs are very important for both parties. On one hand, the provider needs to avoid having penalties due to failure of providing the agreed service. On the other hand, the customer favors on-demand service and without any interruptions. The failure of guaranteeing a service, which leads to unwanted consequences such as penalty payments, profit margin reduction, reputation degradation, customer churn and service interruptions is called SLA violations. SLA violations do happen in real world and have caused both providers and customers heavy costs. Hence, it is paramount for providers to predict and prevent SLA violations for BDAAs as much as possible before they happen. However, accurate prediction of SLA violation for BDAAs is extremely challenging with manifold reasons:

- The violation status of BDAA jobs is actually determined by multiple configurations across different layers (see Table 2.1). For example, if a user submits a job with high CPU or memory requests at BDSaaS layer, but BDPaaS layer fails to allocate a reasonable number of instances for batch processing this job (e.g., too small the number of map reduce workers), this could impact considerably the performance of this batch processing job and result in the violation. Therefore, the configurations at each layer are extremely important and determine the violation status of BDAA jobs. However, the hidden pattern between these multiple configurations across BDAA layers regarding the status of SLA violation is far from clear and needs to be well uncovered and utilized.

- In the real world of SLA management in BDAAs, the SLA violation is very rare event, which results in data skewness meaning that the number of violated jobs are much less than the number of unviolated. This could cause deficient classification models and bring lots of challenges for most supervised learning algorithms because they tend to minimize loss by labeling every sample with the majority class(es), leading to poor recall on the minority class(es). However, true misclassification costs may be much greater when minority class instances are missed. E.g., incorrectly predicting the actual "violated" jobs will cause the degradation of reputation and loss of profit for providers.

The question of SLA violation detection has been addressed previously by using machine learning-based approaches in recent years.

Leither et al. [349] proposed an approach to predict SLA violations in runtime mode for compound web services. They built a regression-based prediction model that takes typical service quality data and process instance data as input, such as availability, system workload, response time, ordered products and customer identifiers. They implemented their prediction model by using a fully Java-based machine learning toolkit called WEKA. However, this toolkit is not scalable to suit the situation in real world where the scale of the dataset is much bigger compared to the one which is used in this paper. Also, the layered big data analytics application framework is fundamentally different from their composite web services.

Jules et al. [350] proposed a Bayesian Network-based model that can calculate and constantly update the reputation of a trusted provider. Moreover, they introduced a probabilistic ontology-based technique that can forecast SLA violations in terms of contract parameters. Although their approach achieves decent performance, the dataset is artificially produced by simulation, which has defect in representing a real application workload. For example, SLA violations occupy 40% in their generated dataset, which disregards the reality that SLA violations are very rare ($< 10\%$) in real application deployment scenarios.

The authors in the paper [351] focused on predicting SLA violations for cloud services. They built a prediction model that uses Naive Bayesian algorithm and takes service quality datasets regarding historically measured web service as the input. In this paper, they investigated and validated the most valid feature combinations for prediction. Still, their web application workloads are fundamentally different from big data analytics application workloads and hence is not applicable to predict SLA violations for cloud-hosted BDAAs.

Hemmat et al. [352] systematically compared the performance of two machine learning classification models for predicting SLA violation by analyzing Google cluster trace dataset [353]. They explored several methods of handling unbalanced data. Despite its good performance, the authors resampled the training data before the cross validation, which leads to the problem of information leakage and overfitting. Also, the features they extracted are task-oriented rather than application as a whole. Hence, the prediction outcome of SLA violation is actually for tasks, which is not what a provider really cares about.

Uriarte et al. [354] performed their SLA violation approach using Google Cluster trace dataset as well. They clustered the resource usage and duration of services using an unsupervised learning-based technique. If a service in a cluster is detected as a violation, the prospective resources will be allocated to the other services hosted in the same cluster in order to avoid the further violation. Although it is helpful for avoiding violation in this cluster, explicit violation forecast towards each service is lacking.

To the best of our knowledge, the above works have the following limitations.

- The above works mainly investigated SLA violation regarding web service, or general cloud service while ignoring application types
- None of them systematically investigated and compared different machine learning algorithms with the integration with diverse skewness handling techniques (e.g., oversampling, undersampling or combined) under various evaluation metrics (e.g., accuracy, precision and recall). Thus, it is not clear (yet) for providers on how to choose the best performing combination of machine learning predictors and skewness handling techniques
- The above approaches investigated the prediction of SLA violation only by exploring the performance (e.g., accuracy, precision and etc.,) of predictors, while ignoring the relationship between the ratio of SLA violation and providers' profit.

7.2 Problem Formulation of SLA Violation Detection

We formulate the detection of SLA violation for cloud-hosted BDAAs as a binary classification problem. Firstly we define the notations to describe this problem in Table 7.1.

Given a set of features extracted from the Alibaba cloud trace (see Section 7.3), what is the probability of failure of a submitted batch job, which results in SLA violation, or the probability that this batch job is successfully processed without SLA violation?

A binary classification problem aims to find a function ϕ: $X \rightarrow Y$ based on D_n that gives a new sample $X_{new} \in X$, predicts $\hat{Y} = \phi(X_{new}) \in Y$.

7.3 Dataset

In this thesis, we pursue our exploration on a BDAA workload dataset released by Alibaba in September 2017 [355]. As a leading public cloud platforms in the world, Alibaba Cloud is running millions of batch jobs or online services across hundreds of datacenters every day using the latest big data technologies. This dataset contains

Table 7.1: Notation used in formulating SLA violation detection problem

Symbols	Description
X	Represents a space containing descriptions of batch jobs. $X = (X_1, X_2, ..., X_m)$, where X_m statistically describes the property of a batch job from an aspect, such as the requested CPU configuration (e.g., speed, number of cores), or requested memory and so on
Y	Represents a space labeling SLA violation status. For the binary problem, there are only two classes. Y = [0, 1]. Y=0 means the batch job is unviolated and Y=1 means the batch job is violated
n	The number of observed samples
D_n	Denotes the training dataset. $D_n = [(x_1, y_1), ..., (x_i, y_i), i = 1, 2, ..., n] \in (X, Y)$

a workload trace of a BDAA cluster. Each trace file includes the job statistics over 12 hours and has information about 1300 virtual machines that execute both offline batch jobs and online services in this cluster. As regards to the batch processing BDAA workloads, the trace details the information including job ids, task ids, instances types, and machines' hardware configuration. To the best of our knowledge, this dataset has not been extensively utilized by the research community. Lu et al. [356] performed characterization of the Alibaba cloud trace and disclosed four types of imbalance (i.e., Spatial imbalance, Temporal imbalance, Proportion imbalance of resources utilization per workload, and Resource demands and runtime statistics imbalance). The work in [357] investigates the elasticity and plasticity of resource allocation of the Alibaba trace. The authors in [358] focused on providing a unique and microscopic view about how the co-located workloads interact and impact each other.

In the Alibaba dataset, three types of information are available with regards to job deployment and execution, including machine utilization, runtime batch processing job workload information, and runtime online service workload. For the sake of confidentiality, Alibaba has excluded or obfuscated part of the dataset. For instance, they normalized the values regarding the utilization of disk and memory. For every

machine, service workload is given a numeric id, which is unique in the trace period. No service and task names are given. In this thesis, we focus on the batch workload information whose entity relationship diagram is shown in the Figure 7.1.

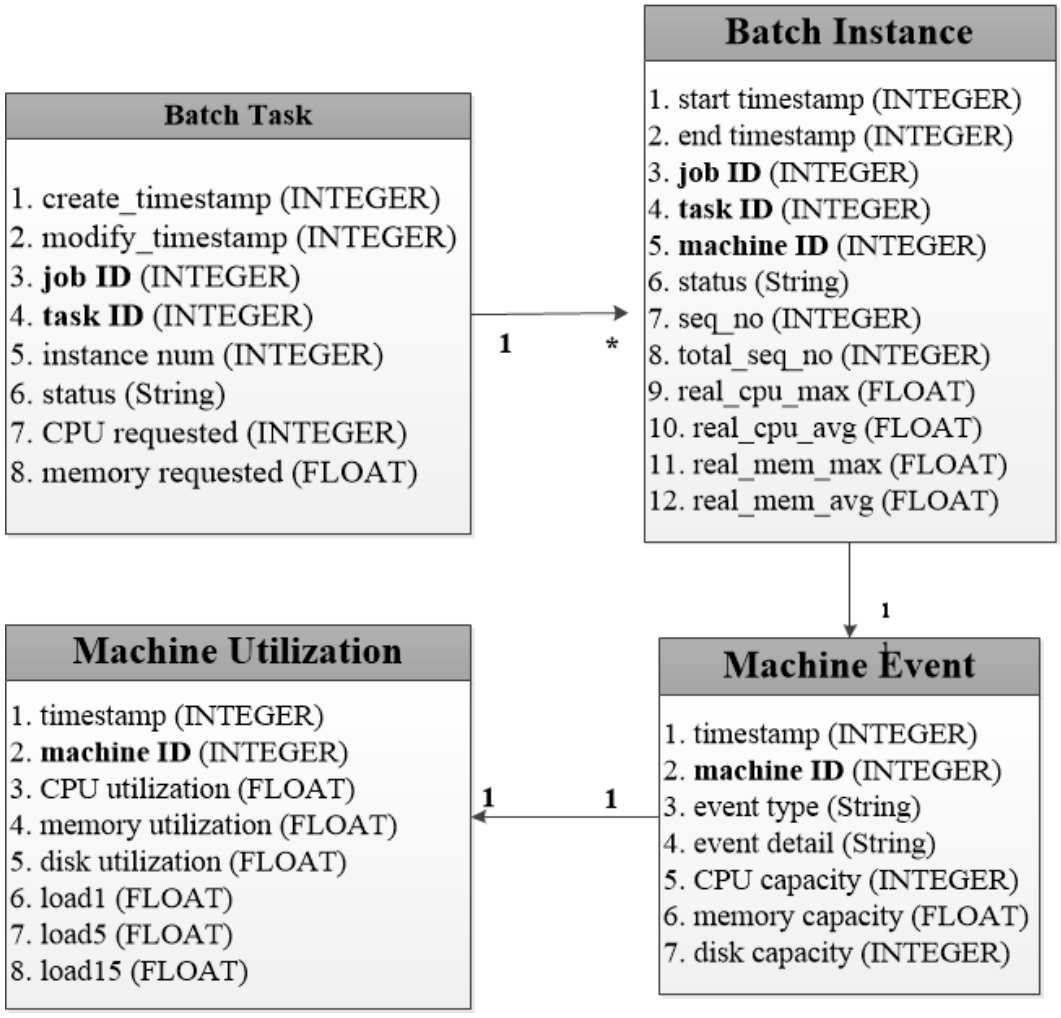


Figure 7.1: Entity relationship diagram of batch workload information in the Alibaba datase

It is seen that batch workloads are depicted in two tables: "Batch Task" and "Batch Instance" table. A batch workload is submitted by the user in the form of a job. Each job comprises various tasks that were submitted to the cluster and forms a directed acyclic graph (very similar to MapReduce execution graph) according to the data dependency. The event cycle of the tasks inside each job is traced in the

"Batch Task" table. Each task consists of numerous instances and executes diverse computing logics. The instance is the smallest scheduling unit of the batch workload in Alibaba's cloud platform. All instances inside a task process the identical binary code with same multiple resource demands but executes different fragments of data. Such execution flow is in line with the MapReduce programming model. Each instance within a task pertains to one job and is then assigned to a specific cluster computing machine that utilizes a Linux container to execute that task instance. While a majority of batch processing jobs can include hundreds of tasks instances, few selected ones can also include an extremely large number of task instances, for example, work in [356] reports that a job had 60000+ task instances. In our research, the metadata related to task instances is tracked in "Batch Instance" table. The meta information of machine in an Alibaba cluster is captured in the following two tables ("Machine Events" and "Machine Utilization" table). The "Machine Events" table shows three types of events (addition, soft error or hard error), and reflects the normalized physical capacity of each machine in terms of RAM size and CPU cores. "Machine Utilization" table records the attributes of each machine such as machine ID, utilization of CPU, utilization of memory and so on.

The configurations of job, task, instance, and machine are critical elements for batch workloads in BDAAs across different layers. The structure of batch workloads running on Alibaba cluster machines is demonstrated in Figure 7.2.

7.4 Feature Extraction

Before we apply machine learning techniques to detect SLA violation, it is essential to extract features based on the batch workload information in this dataset. Since a job consists of multiple tasks, for generality, let a job $J = (T_1, ..., T_i, ..., T_m)$. For each task T_i in J, it consists of multiple instances. Let $T_i = (Inst_1^i, Inst_2^i, ..., Inst_k^i, ..., Inst_N^i)$ where N_i denotes the number of instances of T_i always taking an integer value ≥ 1.

Let $Inst_k^i(real_cpu_max)$ denotes the maximum CPU numbers of actual instance running for the task instance, $Inst_k^i(real_cpu_avg)$ denotes the average CPU numbers of actual instance running for the instance, $Inst_k^i(real_mem_max)$ denotes the

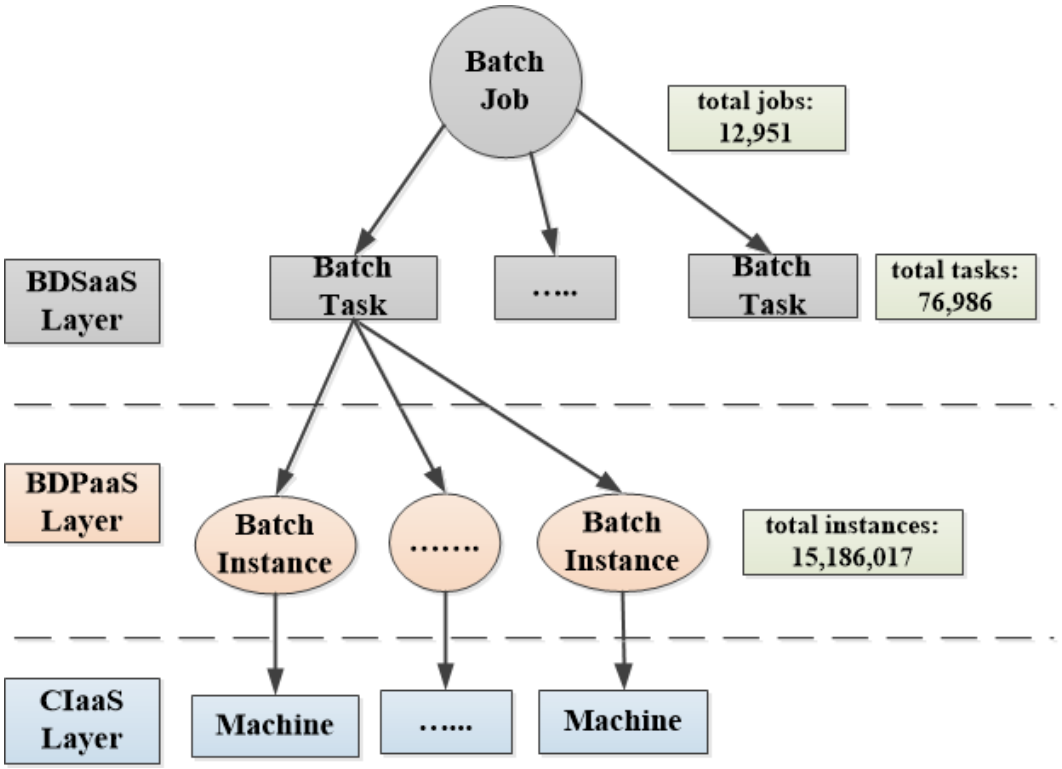


Figure 7.2: The structure of batch workload in Alibaba trace dataset

maximum normalized memory for the instance, $Inst_k^i(real_mem_avg)$ denotes the average normalized memory for the instance, $Inst_k^i(cpu_capacity)$ denotes the normalized physical CPU capacity of the machine that has been used by the instance, $Inst_k^i(mem_capacity)$ denotes the normalized physical memory capacity of the machine that has been used by the instance , $T_i(cpu_requested)$ denotes the CPU requested for each instance of the task, and $T_i(mem_requested)$ denotes the normalized memory requested for each instance of the task. Then, for each job J, we do a set of mathematical aggregation operations on the above variables in order to extract features across layers. Finally, ten features are generated as follows.

BDSaaS layer:

- $X_1 : cpu_requested_per_job = \dfrac{\sum_{i=1}^{m} T_i(cpu_requested)}{m}$

- $X_2 : memory_requested_per_job = \frac{\sum_{i=1}^{m} T_i(memory_requested)}{m}$
- $X_3 : num_tasks_per_job = m$

BDPaaS layer:

- $X_4 : num_instances_per_job = \frac{\sum_{i=1}^{m} N_i}{m}$
- $X_5 : real_cpu_max_per_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(real_cpu_max)/N_i}{m}$
- $X_6 : real_cpu_avg_per_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(real_cpu_avg)/N_i}{m}$
- $X_7 : real_mem_max_per_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(real_mem_max)/N_i}{m}$
- $X_8 : real_mem_avg_per_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(real_mem_avg)/N_i}{m}$

CIaaS layer:

- $X_9 : cpu_capacity_per_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(cpu_capacity)/N_i}{m}$
- $X_{10} : mem_capacity_per_job = \frac{\sum_{i=1}^{m}\sum_{k=1}^{N_i} Inst_k^i(mem_capacity)/N_i}{m}$

As high level features, the above ten features $(X_1 - X_{10})$ provide an effective representation of each batch job J. We could also consider other criterion such as Linux CPU load, disk space requested, or running trials number and etc., as features. However, we refrain from using redundant features for the purpose of preventing the model suffering from overfitting.

7.5 SLA Violation Definition

Besides the features, we also need to formulate the target SLA violation status [0: "non-violated", 1: "violated"] for each batch J. According to the dataset, there are four main types of states for a batch task:

- Terminated: a task goes to Terminated when all its instances are done, meaning this batch task successfully processed
- Waiting: a task is not initialized yet
- Failed: a task fails
- Running: a task is being processed

In this dataset, most of tasks are terminated in a normal mode, while 2000 plus have "waiting" state. Since a job consists of multiple tasks, we can set the criteria that for one job, only when the state of all of its tasks is "Terminated", the job is regarded as "Terminated". If one of its task's state is "Failed", the job is regarded as "Failed". Similarly, if one of its task' state is "Running", the job is regarded as "Running".

There are 12951 jobs in total. We have performed data quality check and pre-processing work as the foremost step. We removed some "abnormal" jobs whose corresponding tasks have missing values in the dataset, and then dropped some "abnormal" jobs whose latest finished task is earlier than its earliest created tasks. After this, 11897 valid jobs remain, where 10774 jobs are labeled "Terminated" (successfully processed), and 1123 jobs are labeled "Failed" based on the above criteria.

In order to determine SLA violations, we need to have particular details in terms of the service quality and service level objectives. Although such information is not available in this dataset, we could discover SLA violation of a job according to the availability of its tasks. Specifically, we detect a job as "violated" if at least one of its corresponding tasks is failed and unusable, representing this job failure causing loss for both providers and customers. Accordingly, we detect a job as "non-violated" if all tasks of this job are terminated normally, representing that this job is successfully finished.

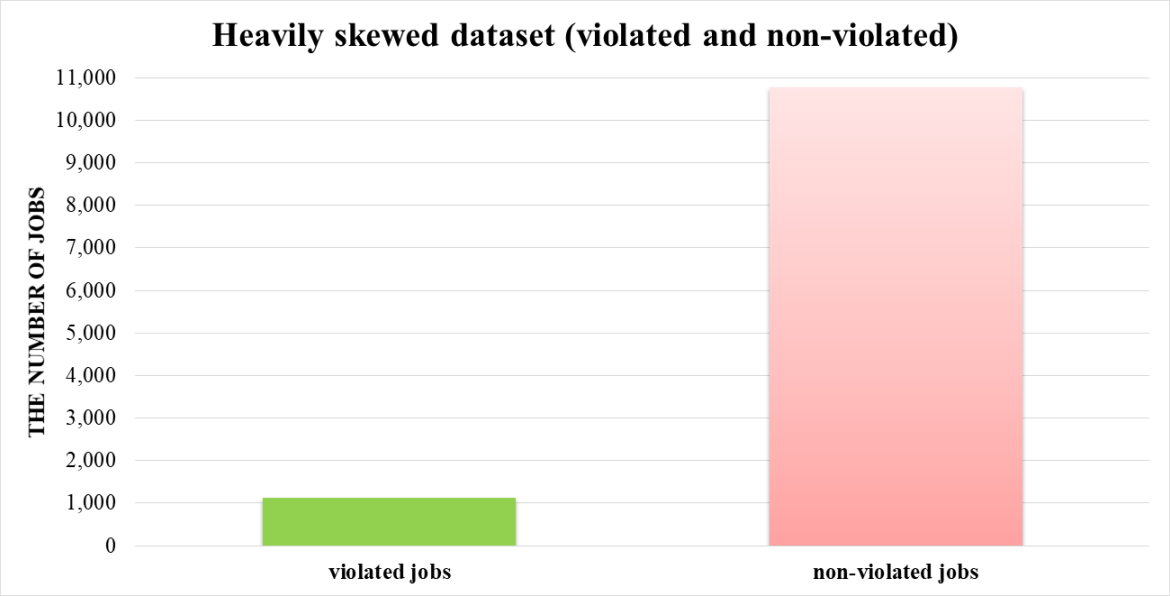


Figure 7.3: Heavily skewed dataset (violated and non-violated)

Based on this SLA violation definition, the percentage of "non-violated" batch jobs (10774 in total) among all the valid jobs (11897 in total) is 90.56%, and hence the ratio of SLA violation is only 9.44%. We can conclude that the quantity of violated and non-violated jobs are highly skewed, which is visualized in Figure 7.3.

7.6 Examples of Features and SLA Violation Status

For the ten features ($X_1 - X_{10}$), their feature ID, feature name, description, type and corresponding layer are detailed in Table 7.2. Further, Table 7.3 gives specific samples of two classes (non-violated and violated) in terms of these feature values.

It is observed that the range of values in the above table varies widely, such as the value of feature X_2 in the violated example is 0.0055, while the value of feature X_1 in the violated example is 100. This is because, in this dataset, the values for disk and memory utilization have been normalized for confidentiality reasons, while the values for requested CPU have not been normalized.

It is worth noting that the absence of normalization will cause objective functions of most machine learning algorithms work improperly. This is because the majority of

Table 7.2: Summary of the ten features and their classification by layer

Layer	Feature ID	Feature Name	Description	Type
BDSaaS	X_1	*cpu_requested_per_job*	A floating point number indicating the amount of requested CPU	Float
	X_2	*mem_requested_per_job*	A floating point number indicating the normalized amount of requested memory	Float
	X_3	*num_tasks_per_job*	An integer representing how many tasks a batch job has, with one as the minimum number of tasks	Integer
BDPaaS	X_4	*num_instances_per_job*	A floating point number indicating the number of instances per job, with 1.0 as the minimum number of instances	Float
	X_5	*real_cpu_max_per_job*	A floating point number indicating maximum CPU numbers of actual instance running	Float
	X_6	*real_cpu_avg_per_job*	A floating point number indicating average CPU numbers of actual instance running	Float
	X_7	*real_mem_max_per_job*	A floating point number indicating maximum normalized memory of actual instance running	Float
	X_8	*real_mem_avg_per_job*	A floating point number indicating average normalized memory of actual instance running	Float
CIaaS	X_9	*cpu_capacity_per_job*	A floating point number indicating the capacity of CPU of a machine	Float
	X_{10}	*mem_capacity_per_job*	A floating point number indicating the capacity of normalized memory of a machine	Float

Table 7.3: Samples of two classes (violated and non-violated Job)

Feature	Feature Description	Violated Job	Non-Violated Job
X_1	*cpu_requested_per_job*	100	50
X_2	*mem_requested_per_job*	0.0055	0.0094
X_3	*num_tasks_per_job*	7	14
X_4	*num_instances_per_job*	85.57	87.86
X_5	*real_cpu_max_per_job*	0.7819	4.1895
X_6	*real_cpu_avg_per_job*	0.4324	0.2622
X_7	*real_mem_max_per_job*	0.0083	0.0143
X_8	*real_mem_avg_per_job*	0.0057	0.0098
X_9	*cpu_capacity_per_job*	63.5719	63.8789
X_{10}	*mem_capacity_per_job*	0.6853	0.6886

classifiers in these algorithms use Euclidean distance method to calculate the distance between two points. The value of distance will be dominated by a particular feature that has a broad range of values. Therefore, we normalize all the above features (from X_1 to X_{10}) by applying standardization methods such as Min-Max scaling [359], so that each feature has approximately proportionate contribution to the final Euclidean distance.

7.7 Prediction Models

Many machine learning prediction models could be explored to detect SLA violation in the Alibaba dataset. However, we particularly selected Logistics Regression (LR) [360], Artificial Neural Network (ANN) [361], Random Forest (RF) [362] and Extreme Gradient Boosting (XGB) [363] for the following reasons:

- LR is one of the elementary and likely most commonly used machine learning algorithms for solving all classification problems. It is easy to implement, fast to train and returns probability scores.
- ANN is one of advanced machine learning models leveraging deep learning technology. It works by splitting the problem of classification into a layered network of simpler elements. Hence, it is very meaningful to apply ANN to detect SLA violation for layer-based big data analytics applications.
- RF is one of popular ensemble machine learning models. It is an improvement over bagged decision trees and returns the importance of all features, which provides insightful information regarding features contribution to SLA violation detection.
- XGB is also one of ensemble machine learning models and returns the importance of all features. Unlike RF, it is based on boosting concept.

7.7.1 Logistics Regression

According to [360], the violation probability can be expressed as a function of the covariates $X_1, ..., X_n$. We denote this probability as $\phi(x_1, ..., x_p)$ for $x_i \in X_i$. We could

consider a linear model for the function ϕ:

$$\phi(x_1, ..., x_n) = \beta_0 + \beta_1 * x_1 + ... + \beta_n * x_n$$

However, the linear equation can yield values in the range $(-\infty, \infty)$, while the probability function ϕ can only take values in the range (0, 1). This problem can be solved using a logit transformation. The formula of this transformation for a given probability p is specified below.

$$(p) = log(\frac{p}{1-p})$$

Applying this transformation on the left side of the equation, we obtain a logistic regression model:

$$(\phi(x_1, ..., x_n)) = \beta_0 + \beta_1 * x_1 + ... + \beta_n * x_n$$

7.7.2 Artificial Neural Network

An artificial neural network (ANN) is a computational model that is inspired by the way biological neural networks in the human brain process information.

Neuron, often called a node or unit, is the basic unit of computation in a neural network. It receives inputs from some other nodes, or an external source, and computes an output. Each input has an associated weight (w), which is assigned by its relative importance to other inputs. The node applies a function f to the weighted sum of its inputs shown in Figure 7.4.

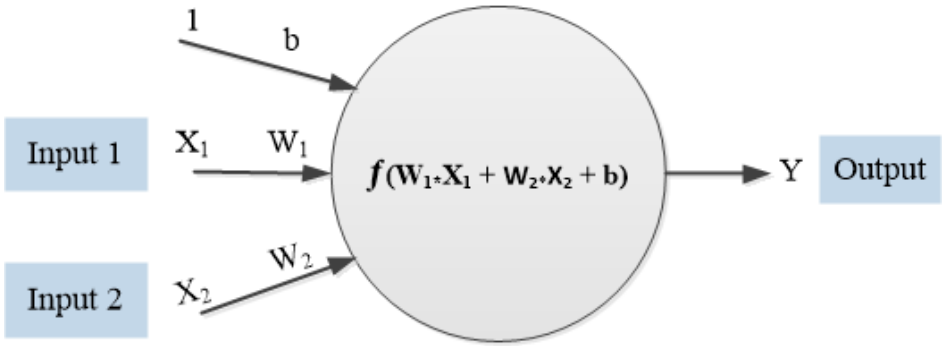


Figure 7.4: Computation and transformation process of a single neuron

The above network takes numerical inputs X_1 and X_2 and has weights w_1 and w_2 associated with those inputs. Additionally, there is another input 1 with weight b (called

the bias) associated with it. The output Y from the neuron is computed as follows:

$$Output\ of\ neuron = Y = f(w_1 * X_1 + w_2 * X_2 + b)$$

The function f is non-linear and is called the activation function. The purpose of the activation function is to introduce non-linearity into the output of a neuron. This is important because most real-world data is non-linear and we want neurons to learn these non-linear representations. Take the Alibaba dataset as an instance, the relationship between the output value (0 denotes non-violated, and 1 denotes violated) and the ten features inputs are not as simple as linearly represented but much more complicated. Hence the activation function f is applied to learn the hidden non-linear pattern between them.

The feed-forward neural network was the first, and the simplest type of artificial neural network devised [364]. It contains multiple neurons (nodes) arranged in layers. Nodes from adjacent layers have connections or edges between them. All these connections have weights associated with them.

In this thesis, we designed and built a 3-layer feed forward neural network. It contains ten neurons (to be consistent with the number of features stated in Section 7.4) in the input layers, three neurons (to be consistent with the three layers) at the hidden layer, and one neuron at output layer, which is shown in Figure 7.5.

7.7.3 Random Forest

Random Forest (RF) was firstly introduced by Breiman extending previous research on random decision trees and bagging of predictors [362]. Briefly, RF consists of multiple single decision trees. Each tree is a fairly simple model that has branches, nodes, and leaves. The nodes contain attributes on which the objective function depends on. Then the values of the objective function are mapped to the leaves through the branches. In the process of classification of a new case, it is necessary to go down the tree through branches to a leaf, passing through all attribute values according to the logical principle "IF-THEN". Depending on these conditions, the objective variable will fall into a particular leaf, representing a particular value or a class that this case belongs to.

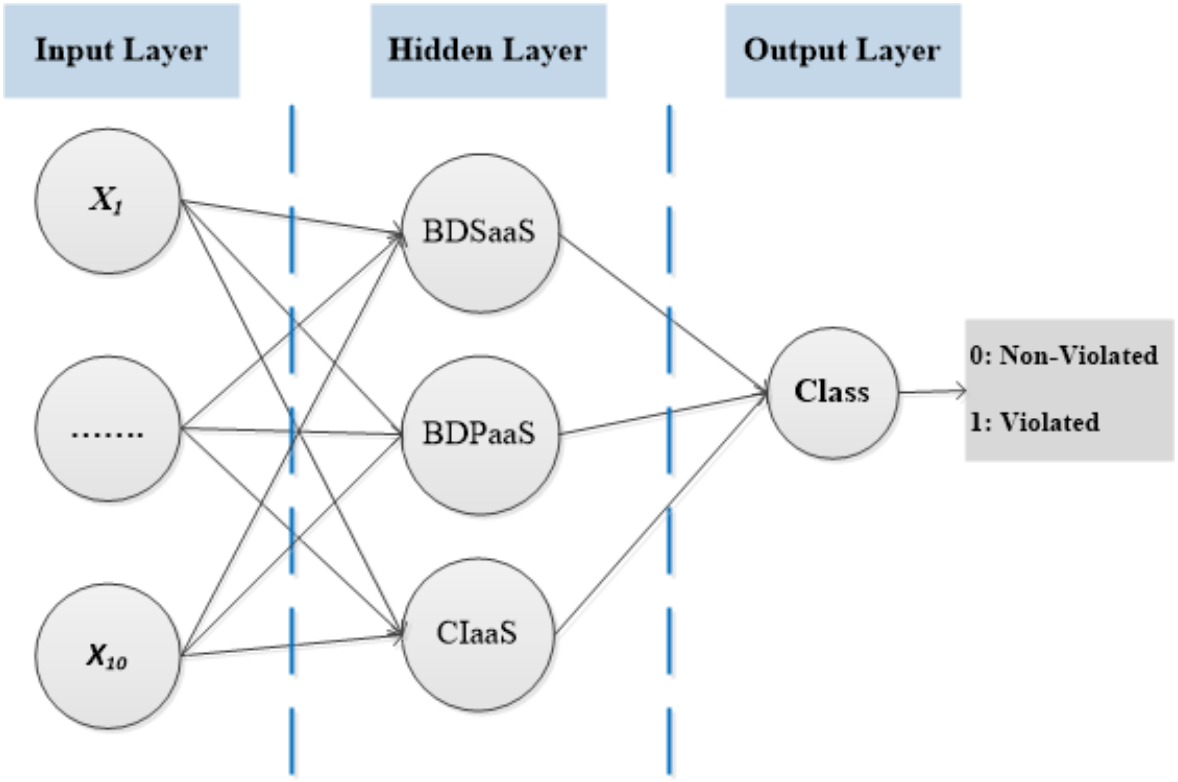


Figure 7.5: Design of a 3-layer neural network for Alibaba trace dataset

A core methodology leveraged by RF is a bagging-based ensemble learning. Bagging is an abbreviation of bootstrap aggregating. The conventional bagging algorithm involves generating n different bootstrap training samples with replacement, and training the algorithm on each bootstrapped algorithm separately and then aggregating the predictions at the end. Bagging is used for reducing overfitting in order to create strong learners for generating accurate predictions. Bagging allows replacement in the bootstrapped sample. It can be said that a random forest is a special case of bagging, where decision trees are used as the base family. Figure 7.6 gives a pictorial representation of RF's working mechanism.

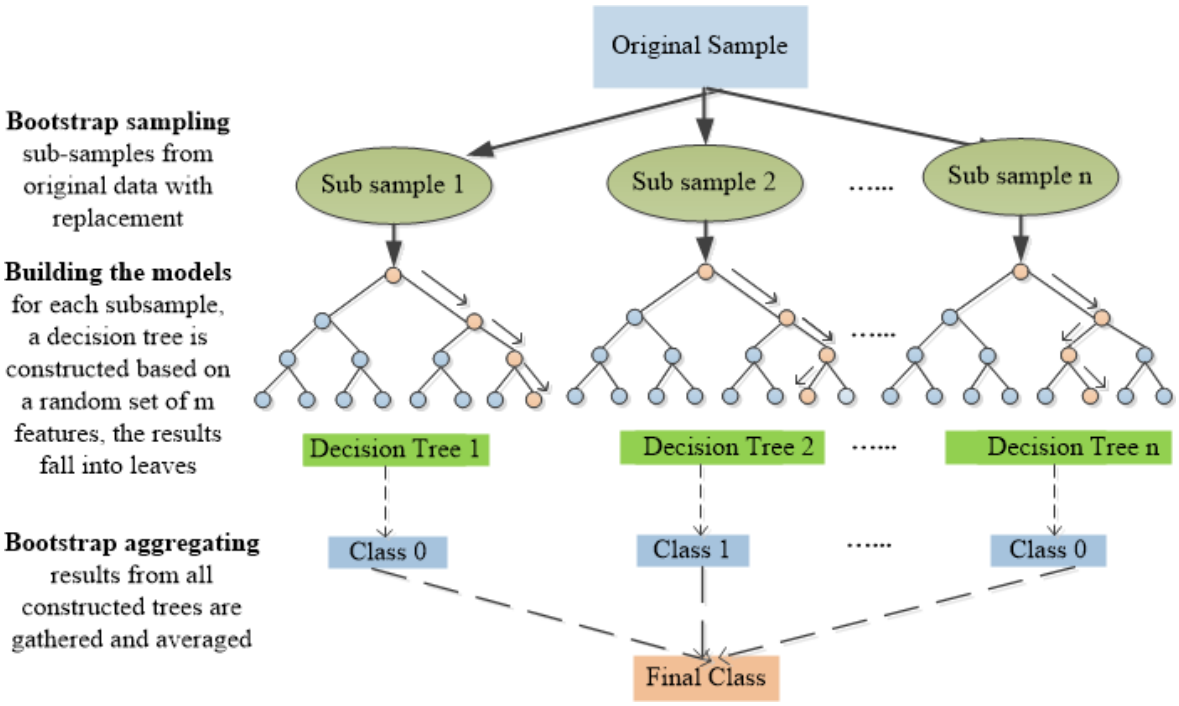


Figure 7.6: Working mechanism of random forest

7.7.4 Extreme Gradient Boosting

Unlike RF, extreme gradient boosting (XGB) is a boosting-based ensemble learning method and an advanced and more efficient implementation of Gradient Boosting Algorithm [363, 365]. Boosting is an ensemble technique to combine weak learners to create a strong learner that can make accurate predictions. Decision Trees are used as weak learners in XGB. Boosting starts out with a base classifier / weak classifier that is prepared on the training data. The base classifiers are weak learners, i.e. the prediction accuracy is only slightly better than average. A classifier learning algorithm is said to be weak when small changes in data induce big changes in the classification model. In the next iteration, the new classifier focuses on or places more weight to those cases which were incorrectly classified in the last round. Figure 7.7 gives a pictorial representation of XGB's working mechanism.

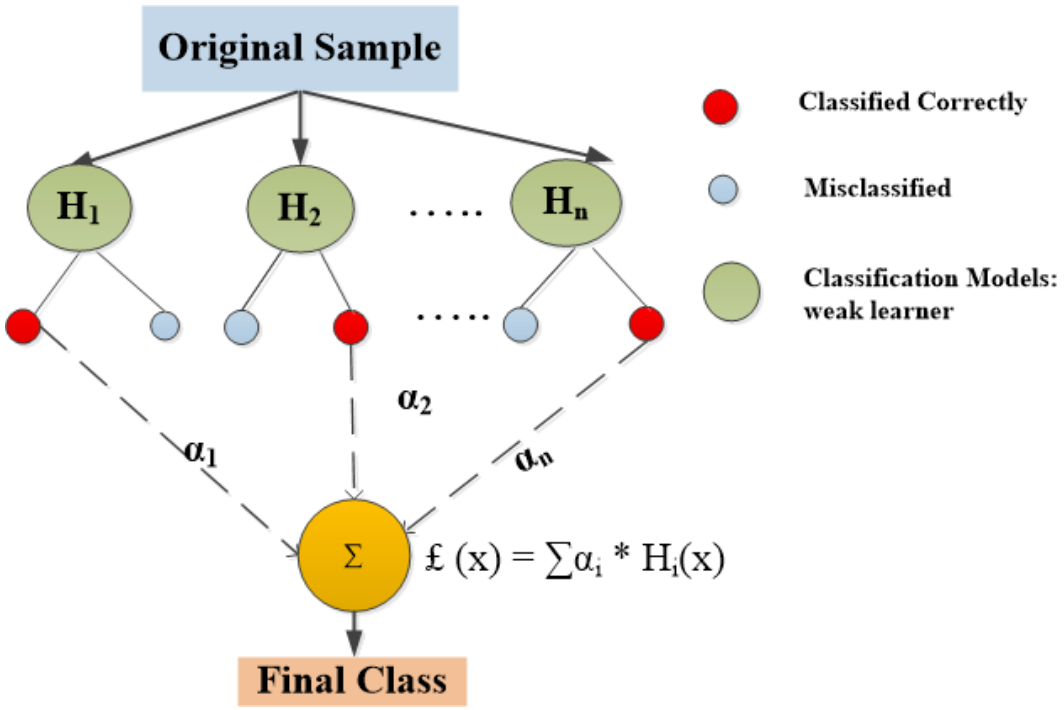


Figure 7.7: Working mechanism of XGB

7.8 Tackling Unbalanced Data

We feed the dataset to the aforementioned four machine learning prediction models to detect future SLA violations. Since the two classes [0: "non-violated", 1: "violated"] in the Alibaba dataset are heavily unbalanced, this is known as data skewness, which makes the classification task extremely hard, because the classifier will always tend to predict the dominant class. However, true misclassification costs may be much greater when minority class instances are missed. E.g., incorrectly predicting the actual "violated" jobs will cause the degradation of reputation and loss of profit for providers.

As a broadly adopted approach in handling unbalanced datasets, resampling efficiently changes the distribution training data for the purpose of biasing the classifier towards the minority class. Basically, resampling can be divided into two groups (i.e., undersampling and oversampling). Undersampling denotes removing samples from the majority class, while oversampling represents adding more examples from the minority class. Different forms of resampling techniques are detailed in the following subsections.

The resampling techniques discussed in this thesis are by no means an exhaustive list.

7.8.1 Oversampling Techniques

7.8.1.1 Random Oversampling)

In random oversampling (ROS), the samples of the minority class are randomly picked and duplicated. The method continues to create duplicate data points until the two classes have roughly the same number of data points [366].

7.8.1.2 Synthetic Minority Oversampling Technique

Synthetic Minority Oversampling Technique (SMOTE) is an oversampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by oversampling with replacement. According to the original research paper [367], synthetic samples are generated in the way that taking the difference between the feature vector under consideration and its nearest neighbour, multiplying this difference by a random number between 0 and 1 and adding it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general.

7.8.1.3 Adaptive Synthetic

Adaptive Synthetic (ADASYN) is based on the idea of adaptively generating minority data samples according to their distributions using K nearest neighbour algorithm. ADASYN adaptively updates the distribution, and there are no assumptions made for the underlying distribution of the data. The algorithm uses Euclidean distance for the KNN algorithm. As a result, ADASYN improves learning concerning the data distributions in two ways: (i) reducing the bias introduced by the class imbalance, and (ii) adaptively shifting the classification decision boundary toward the difficult examples [368].

7.8.1.4 Borderline-SMOTE

Borderline-SMOTE is a variant of SMOTE that generates synthetic samples only on the border line of the two classes [369]. The data points which are in the borderline of the classes are more prone to miss-classification and thus need more attention. It is based on SMOTE for generating new data points. The synthetic data points are created on the border line and between the minority data points and their selected nearest neighbours.

There are two types of Borderline-SMOTE algorithm including Borderline-1 and Borderline-2 with the parameters kind ="borderline-1" and kind = "borderline-2". In Borderline-1 SMOTE, the generated data points will belong to the same class as the one of the sample data points. On the contrary, Borderline-2 SMOTE will consider the generated data points which can be from any class.

7.8.2 Undersampling Techniques

7.8.2.1 Random Undersampling

Random undersampling (RUS) is the case of randomly deleting data points from the dominant class until both classes have roughly the same size. This method might delete the data points in the decision boundary that are important in the process of decision making [366].

7.8.2.2 One-sided Selection

One-sided Selection (OSS) uses a combination of Tomek links and an extension of Nearest Neighbor, which is defined as an optimization problem finding closest points [370]. One-sided Selection finds the safe samples and removes the unsafe samples from the majority class. Tomek links removes the samples near the border line, and Nearest Neighbor removes the samples that are far from the border line.

7.8.2.3 NearMiss

NearMiss 1, 2 and 3 algorithms [371] are undersampling methods. NearMiss 1 removes the data points from the majority class, whose average distance to three closest

data points in the minority class is the smallest. NearMiss 2 chooses the majority class data points, whose average to all data points in the minority class is the smallest. NearMiss 3 removes a given number of majority class data points for each data point in the minority class. The selection is made by applying a clustering and selecting a sample from each clustered neighborhood.

7.8.3 Combination of Oversampling and Undersampling Techniques

7.8.3.1 SMOTE-Tomek Links

Since SMOTE oversampling might lead to overfitting and Tomek links undersampling might remove important data points, the ensemble of these two methods provides better results. In SMOTE-Tomek links [372], we first over-sample the minority class with SMOTE and then under-sample using Tomek links both the majority and the minority classes in producing a more balanced dataset.

7.8.3.2 SMOTE-ENN

SMOTE-ENN [372] is also a combination of SMOTE and ENN. SMOTE is used as the over-sampler for minority class, and then ENN provides data cleaning for both classes.

7.9 Evaluation Metrics

In order to measure the performance of the above prediction models in the dataset, evaluation metrics are required, which helps us indicate how skilfully a model will perform. Thus, after a prediction model is trained on the training set, it must be validated on an unseen testing dataset. This approach is beneficial in choosing a model that will have reasonably decent performance on unknown dataset. In this thesis, five diverse evaluation metrics have been used. They include Accuracy, Precision, Recall, Receiver Operating Characteristic (ROC) area, and F_β score.

Let us first present the confusion matrix in Table 7.4 which will help us to define the

Table 7.4: Confusion matrix in binary classification

	Positive (Real)	**Negative (Real)**
Positive(Prediction)	True Positive (TP)	False Positive (FP)
Negative(Prediction)	False Negative (FN)	True Negative (TN)

above metrics. We can see that in a confusion matrix, the correctly classified instances are in the diagonal of the matrix, the True Positive (TP) and True Negative (TN) cases. The misclassified instances are the False Positive (FP) and False Negative (FN) ones. In our problem, FP denotes the quantity of examples that are erroneously classified as "violated" where the real label should be "non-violated". Similarly, TP represents the number of examples that are correctly classified as "violated" where the real label should be "violated". Regarding SLA violation detection, it is worth noticing that the most important value to increase is the number of TPs cases, which correspond with the correctly detected SLA violations. Metrics involving the TNs are usually not useful because this number is usually much higher than its TP counterpart, as SLA violations are rare events. Therefore, our objective is to find the right balance of FNs and FPs, while maximizing the TP observations. Usually, minimizing the FN instances is prioritized over minimizing the FPs, due to the higher value in detecting new SLA violations and higher loss that new SLA violations cause. Based on the above confusion matrix, different metrics can be computed depending on what we are interested in measuring (details of these evaluation metrics are described in the following subsections).

7.9.1 Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

It is worth noting that accuracy can be misleading. In a dataset with highly unbalanced classes, if the classifier always "predicts" the majority class without performing any analysis of the features, it will still have a high accuracy rate. Obviously this quality parameter should be considered illusory.

7.9.2 Precision

$$Precision = \frac{TP}{TP + FP}$$

The precision is a metrics to interpret the capability of a classifier how many are actually "violated" out of the jobs that the classifier predicted to be "violated". Its value is in the range [0, 1] with 0 being the worst, and 1 being perfect.

7.9.3 Recall

$$Recall = \frac{TP}{TP + FN}$$

The recall represents the capability of a classifier how many are found to be "violated" by the classifier out of all the jobs that are actually "violated". The value of recall is also in the range [0, 1] with 0 being the worst, and 1 being perfect.

7.9.4 F_β Score

$$F_\beta \text{ score} = (1 + \beta^2) * \frac{precision * recall}{\beta^2 * precision + recall}$$

In most of the cases, models with high precision suffer from low recall and vice-versa. In order to combine both metrics, the F_β score is used to harmonize precision and recall, which converts them into a value ranging between 0 and 1. If we want to give the same importance to precision and recall, then F_1 is used. However, as we want to prioritize reducing the FNs, we can use the F_2, which favors maximizing the recall.

7.9.5 Receiving Operating Characteristics (ROC)

Receiving Operating Characteristics (ROC) curve [373] is a curve created by evaluating a classifier over a range of different thresholds. This threshold refers to the discrimination threshold of a binary classifier. In our SLA violation detection problem, the output of a classifier will indicate the probability of a job being predicted as "violated". Different prediction outcome can be obtained depending on where we set the probability threshold for considering an SLA violation. In a ROC curve, the x-axis represents the false positive rate (FPR) while the y-axis measures the recall, where FPR is calculated by the formula of $\frac{FP}{FP+TN}$.

The area under the ROC curve is a common assessment to evaluate the quality of a classifier. Using it, we can determine how a classifier performs across all the thresholds, so this metric comprises more information than the ones derived from the confusion matrix, as there are different confusion matrix for different thresholds. The formal definition for the metrics based on the area under ROC curve is [373]:

$$ROC = \int_0^1 (\frac{TP}{TP+FN} * d(\frac{FP}{FP+TN})) = \int_0^1 (recall * d(FPR))$$

7.10 Evaluation and Analysis of the Prediction Models

In order to train and validate the skill of the four prediction models on unknown data, a 10-fold stratified cross validation has been used on 11897 batch jobs obtained from the Alibaba trace.

In each fold, the original dataset has been randomly split into two parts. One part is used for training phase while the remainder is used for validation phase. The ratio of "non-violated" and "violated" jobs in the validation part remains equal with the ratio in the original dataset. To avoid leaking the information of validation data to training data, which often results in overfitting, it is critical to perform the cross validation before resampling. Resampling can only be done on the samples which are applied for training the particular type of machine learning predictor.

The values of accuracy, precision, recall, F_2 score and ROC area for each prediction technique (representing a particular predictor performing either in the original imbalanced dataset or in the resampled dataset using a particular skewness handling technique) are calculated in each fold. Then, their statistics information can be acquired after repeating the experiments ten times. Moreover, in order to compare how skilful the four prediction models perform, we introduced two more classifiers.

Ideal Classifier: represents a classifier that can perfectly predict those actual "non-violated" batch jobs as "non-violated" and those actual "violated" batch jobs as "violated". In this case, all of evaluation metrics (i.e., accuracy, precision, recall, F_2 and ROC area) are 1.0. This is an ideal classifier and could be regarded as the best case.

Baseline Classifier: this classifier applies the simplest rule on our SLA violation detection problem and it simply predicts every sample in the dataset as the majority class (i.e., "non-violated"). It can be derived that the accuracy of Baseline classifier is 0.9056 and the ROC area is 0.5. The value of precision, recall and F_2 score are 0.0. This classifier involves the least effort of prediction and could be regarded as the baseline.

7.10.1 Results of Cross Validation

7.10.1.1 Logistics Regression

It is observed from Table 7.5 that logistics regression (LR) using all of the resampling techniques except one-sided selection (OSS) outperforms LR in the original dataset regarding F_2 score and ROC area. However, none of the above techniques achieves an acceptable F_2 score. The top F_2 score achieved by Borderline-1 is only 59.96%. It is also found that Borderline-1 has the highest ROC value (81.18%) outperforming other techniques. Regarding the recall value, NearMiss-1, NearMiss-2 and Borderline-2 rank the top three, 92.07%, 90.02%, and 89.04% respectively.

In summary, LR's performance is under par with regards to predicting SLA violation on this Alibaba dataset. It means LR is a simple model without the capability of finding the hidden pattern among the multiple configurations across layers. It gives us an indicator that a more complex model should be explored.

7.10.1.2 Artificial Neural Network

It can be seen from Table 7.6 that the prediction outcome is much better than that of logistics regression (LR). The highest recall value is 93.14%, achieved by Borderline-2. Regarding F_2 score, 62% of the above 13 approaches achieve higher than 60%. The top three F_2 score achiever are SMOTE-ENN, Borderline-1 and ROS with 71.7%, 69.66%, and 69.55% respectively. Notably, SMOTE-ENN also ranks the top in the ROC area (87.93%), with an acceptable accuracy value of 85.58%.

It can be concluded that although Artificial Neural Network (ANN) that we designed and implemented is as simple as ten features in the input layer and three neurons in the hidden layer, it achieves a very decent detection of SLA violation in this dataset.

Table 7.5: Cross validation result for logistics regression with default setting

Predictor	**Accuracy**	**Precision**	**Recall**	F_2	**ROC**
LR	91.75% (±0.34)	80.75% (±9.17)	16.65% (±2.93)	19.77% (±3.36)	58.12% (±1.48)
LR+ROS	80.09% (±1.55)	29.08% (±2.29)	76.57% (±4.45)	57.67% (±3.53)	78.51% (±2.52)
LR+SMOTE	80.11% (±1.49)	29.11% (±2.12)	76.57% (±3.79)	57.69% (±3.03)	78.53% (±2.14)
LR+ADASYN	75.56% (±1.53)	26.27% (±1.38)	87.53% (±2.45)	59.65% (±2.04)	80.92% (±1.5)
LR+Borderline-1	75.67% (±1.51)	26.41% (±1.47)	87.97% (±3.79)	59.96% (±2.56)	81.18% (±1.97)
LR+Borderline-2	74.58% (±1.57)	25.68% (±1.42)	89.04% (±3.28)	59.59% (±2.37)	81.06% (±1.83)
LR+RUS	77.73% (±1.85)	26.73% (±1.76)	77.46% (±4.03)	56.07% (±2.43)	77.61% (±1.81)
LR+OSS	91.7% (±0.35)	78.67% (±9.34)	16.65% (±2.9)	19.75% (±3.33)	58.09% (±1.47)
LR+NearMiss-1	45.97% (±2.05)	14.04% (±0.57)	92.07% (±2.45)	43.59% (±1.37)	66.62% (±1.57)
LR+NearMiss-2	43.46% (±8.97)	13.57% (±1.97)	90.02% (±7.4)	41.84% (±1.7)	64.31% (±2.25)
LR+NearMiss-3	72.87% (±4.09)	20.85% (±3.46)	64.91% (±4.28)	45.44% (±4.58)	69.3% (±3.61)
LR+SMOTE-Tomek	80.14% (±1.49)	29.12% (±2.15)	76.48% (±4.0)	57.66% (±3.16)	78.5% (±2.24)
LR+SMOTE-ENN	79.02% (±1.6)	28.28% (±2.04)	79.06% (±3.81)	58.13% (±3.02)	79.04% (±2.18)

In comparison to LR, ANN is much better at understanding the hidden relationship of those multiple configurations across layers and hence leads to better prediction efficiency. It is worth noting that the performance of ANN in the original dataset is unacceptable (recall and F_2 score are only 51.83% and 54.77% respectively), which shows that ANN suffers from data skewness.

7.10.1.3 Random Forest

Based on Table 7.7, it is seen that random forest (RF) outperforms logistics regression (LR) and artificial neural network (ANN). This time, NearMiss-2 wins the highest recall values, which is 95.9%. In comparison with the top recall winner by ANN, RF also improves the recall value by 3% using NearMiss-2. Among the above 13 approaches, 54% of them achieve a F_2 score that is above 80%, which is very encouraging. The top F_2 score is 83.57%, achieved by borderline-2. Compared with the top F_2 score winner

Table 7.6: Cross validation result for a 3-layer artificial neural network

Predictor	**Accuracy**	**Precision**	**Recall**	F_2	**ROC**
ANN	93.86% (±1.03)	75.4% (±4.68)	51.83% (±14.25)	54.77% (±13.61)	75.04% (±6.92)
ANN+ROS	84.57% (±5.9)	38.78% (±7.95)	88.6% (±8.9)	69.55% (±7.58)	86.38% (±4.89)
ANN+SMOTE	84.3% (±3.98)	37.23% (±7.05)	84.77% (±6.33)	66.85% (±5.24)	84.51% (±3.36)
ANN+ADASYN	79.82% (±5.81)	32.2% (±5.9)	92.96% (±2.42)	66.91% (±5.68)	85.7% (±3.43)
ANN+Borderline-1	83.58% (±3.47)	36.31% (±5.14)	91.18% (±1.84)	69.66% (±3.67)	86.98% (±1.86)
ANN+Borderline-2	80.88% (±4.86)	33.18% (±5.55)	93.14% (±2.94)	67.97% (±5.34)	86.37% (±3.1)
ANN+RUS	79.26% (±4.82)	30.51% (±4.32)	87.82% (±5.71)	63.35% (±4.11)	83.1% (±2.67)
ANN+OSS	93.36% (±1.25)	68.16% (±22.97)	43.84% (±18.63)	46.85% (±18.93)	71.18% (±9.01)
ANN+NearMiss-1	47.87% (±7.09)	14.75% (±1.53)	92.97% (±2.27)	44.97% (±2.77)	68.07% (±3.48)
ANN+NearMiss-2	42.03% (±3.11)	13.24% (±0.86)	92.25% (±2.69)	42.04% (±2.2)	64.52% (±2.87)
ANN+NearMiss-3	70.77% (±5.24)	20.21% (±3.46)	67.39% (±7.33)	45.24% (±2.31)	69.26% (±1.91)
ANN+SMOTE-Tomek	85.74% (±3.67)	39.47% (±7.53)	84.15% (±6.28)	68.19% (±6.7)	85.03% (±4.13)
ANN+SMOTE-ENN	85.58% (±2.49)	39.2% (±4.47)	90.83% (±2.77)	71.7% (±3.73)	87.93% (±2.06)

by ANN, RF significantly improved the F_2 score by 17% through the application of borderline-2. Regarding ROC area, SMOTE-ENN achieved 91.95%, ranking the first. Also, SMOTE-ENN has a very high accuracy rate of 95.24%.

It demonstrates that RF is more effective than LR and ANN in predicting SLA violation in this Alibaba dataset. Even without resampling, it still achieves an admissible performance (accuracy = 97.04%, recall = 74.08% and F_2 = 77.22%). RF has better performance because it is a bagging-based tree classifier, which are less sensitive to class distributions, while LR and ANN are very sensitive with the highly biased class distribution and cannot generate any acceptable results without leveraging resampling techniques.

7.10.1.4 Extreme Gradient Boosting

It is shown in Table 7.8 that extreme gradient boosting (XGB) plus NearMiss2 achieved a stunning recall value of 97.15%. In comparison with the top recall score

Table 7.7: Cross validation result for random forest with default setting

Predictor	**Accuracy**	**Precision**	**Recall**	F_2	**ROC**
RF	97.04% (±0.29)	93.28% (±2.46)	74.08% (±3.67)	77.22% (±3.06)	86.76% (±1.78)
RF+ROS	96.81% (±0.43)	87.61% (±3.41)	77.19% (±3.46)	79.03% (±2.97)	88.02% (±1.72)
RF+SMOTE	96.66% (±0.36)	81.61% (±2.69)	83.61% (±3.5)	83.16% (±2.73)	90.82% (±1.69)
RF+ADASYN	95.95% (±0.46)	76.0% (±3.43)	83.79% (±3.6)	82.04% (±2.7)	90.5% (±1.71)
RF+Borderline-1	96.56% (±0.37)	80.85% (±3.67)	83.7% (±3.81)	83.03% (±2.76)	90.8% (±1.77)
RF+Borderline-2	95.99% (±0.39)	75.36% (±3.57)	86.02% (±2.77)	83.57% (±1.64)	91.52% (±1.16)
RF+RUS	93.05% (±0.84)	58.76% (±3.42)	90.2% (±3.91)	81.39% (±2.8)	91.77% (±1.85)
RF+OSS	97.02% (±0.48)	91.69% (±2.71)	75.32% (±4.05)	78.08% (±3.6)	87.3% (±2.06)
RF+NearMiss-1	45.5% (±3.24)	14.08% (±0.77)	93.23% (±2.37)	43.85% (±1.56)	66.88% (±1.79)
RF+NearMiss-2	55.88% (±6.07)	17.39% (±2.09)	95.9% (±1.34)	50.22% (±3.5)	73.8% (±3.4)
RF+NearMiss-3	93.7% (±0.97)	63.49% (±4.72)	80.76% (±3.72)	76.43% (±2.41)	87.91% (±1.55)
RF+SMOTE-Tomek	96.6% (±0.36)	81.81% (±2.99)	82.45% (±3.84)	82.26% (±2.92)	90.26% (±1.82)
RF+SMOTE-ENN	95.24% (±0.59)	69.8% (±3.23)	87.89% (±3.24)	83.51% (±2.56)	91.95% (±1.59)

achiever by random forest (RF) plus NearMiss-2, XGB plus NearMiss-2 improved the recall score by 1.3%. Regarding F_2 score, the performance of XGB plus random oversampling (ROC) (F_2 score = 82.99%) is marginally lower than that of RF plus borderline-2 (F_2 score = 83.57%).

It is also found that without using a resampling technique, the performance of XGB (accuracy = 96.91%, recall = 74.17%, F_2 = 77.07% and ROC area = 96.91%) is very similar with the performance of RF (accuracy = 97.04%, recall = 74.08%, F_2 = 77.22% and ROC area = 86.76%)). Like RF, XGB has better performance because it is also a decision tree-based classifier, which is more tolerable to class distributions. Thus, it achieves decent performance even without resampling.

Table 7.8: Cross validation result for XGB with default setting

Predictor	**Accuracy**	**Precision**	**Recall**	F_2	**ROC**
XGB	96.91% (±0.42)	91.55% (±2.08)	74.17% (±3.97)	77.07% (±3.52)	86.73% (±1.99)
XGB+ROS	93.21% (±0.73)	59.12% (±3.05)	92.43% (±2.99)	82.99% (±1.96)	92.86% (±1.32)
XGB+SMOTE	94.15% (±0.4)	63.59% (±2.09)	89.4% (±3.73)	82.63% (±2.44)	92.02% (±1.7)
XGB+ADASYN	92.11% (±0.57)	54.96% (±2.08)	92.16% (±2.81)	81.13% (±1.83)	92.13% (±1.27)
XGB+Borderline-1	93.45% (±0.53)	60.15% (±2.39)	91.45% (±3.18)	82.78% (±2.18)	92.55% (±1.48)
XGB+Borderline-2	93.2% (±0.61)	59.1% (±2.69)	92.16% (±3.35)	82.81% (±1.96)	92.74% (±1.43)
XGB+RUS	91.57% (±0.78)	53.22% (±2.76)	93.14% (±3.35)	80.9% (±1.79)	92.27% (±1.36)
XGB+OSS	96.92% (±0.38)	90.17% (±1.92)	75.68% (±3.8)	78.16% (±3.27)	87.41% (±1.89)
XGB+NearMiss-1	45.77% (±3.3)	14.27% (±0.76)	94.39% (±1.78)	44.43% (±1.57)	67.54% (±1.94)
XGB+NearMiss-2	41.54% (±2.67)	13.64% (±0.63)	97.15% (±1.43)	43.65% (±1.47)	66.45% (±1.91)
XGB+NearMiss-3	93.15% (±1.25)	60.14% (±5.0)	83.96% (±4.0)	77.69% (±3.69)	89.03% (±2.16)
XGB+SMOTE-Tomek	94.21% (±0.44)	63.81% (±2.19)	89.76% (±3.65)	82.95% (±2.43)	92.21% (±1.67)
XGB+SMOTE-ENN	92.49% (±0.81)	56.34% (±2.92)	92.7% (±3.05)	82.03% (±2.14)	92.58% (±1.41)

7.10.2 Analysis of Prediction Outcome

According to the above extensive experiments results, for each evaluation metrics, we can study how skilful these prediction techniques are when compared with the "Ideal" and "Baseline" classifiers.

According to Figure 7.8, it is observed that among the top 10 accuracy values, only one undersampling technique (one-sided selection) performs well; the remaining are all oversampling techniques. Moreover, two predictors (random forest and extreme gradient boosting) occupy the top ten accuracy list. Specifically, random forest (RF) plus one-side selection (OSS), and extreme gradient boosting (XGB) plus OSS achieve very high accuracy, 97.02%, and 96.92% respectively. However, their accuracy value are gently lower than the accuracy achieved by RF in the original dataset. It is also noted that RF or XGB plus some resampling techniques (one-sided selection, random oversampling, SMOTE, Borderline family, ADASYN and SMOTE - Tomek) significantly

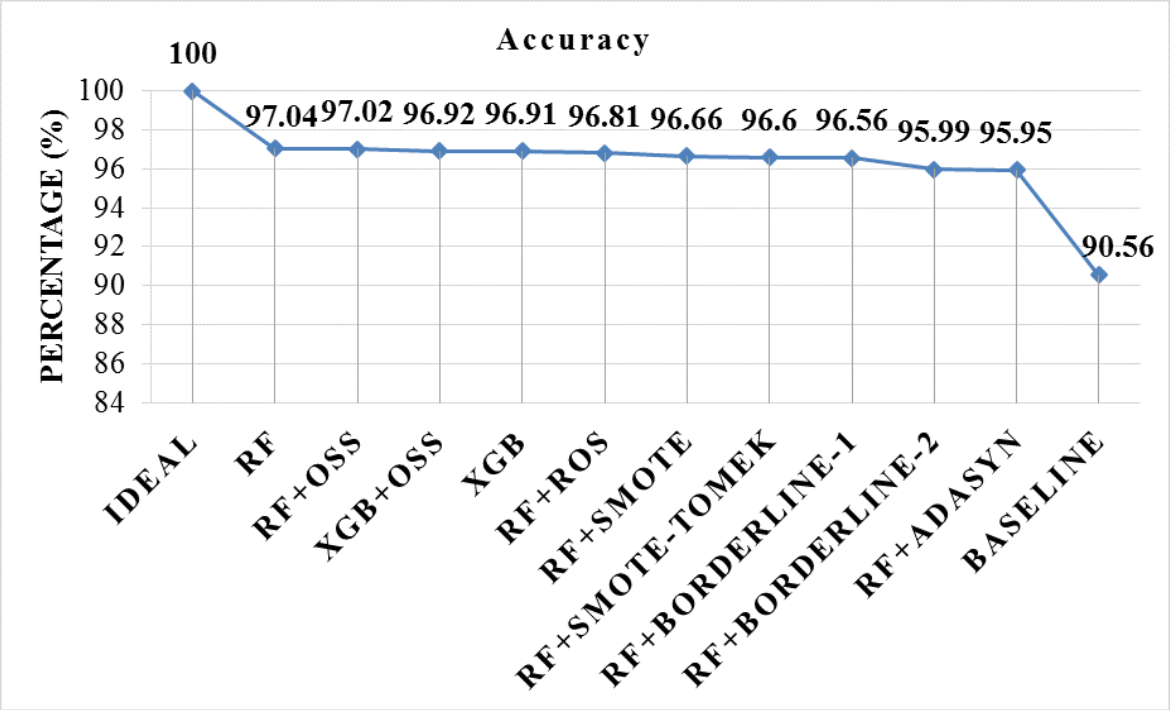


Figure 7.8: Top 10 techniques measured by accuracy

improves the accuracy compared with the Baseline classifier (accuracy = 90.56%). It is concluded that random forest is the technique winning the top accuracy value.

From Figure 7.9, it is found that among the top 10 prediction techniques, only one uses the undersampling technique (one-sided selection); the remaining are all oversampling techniques. One-sided selection (OSS) performs better in achieving high precision compared with other resampling techniques. Take random forest (RF) as an example, the precision value of RF plus OSS is higher than RF plus other resampling techniques. However, the precision of OSS plus OSS (91.69%) is slightly lower than the precision of RF without using resampling techniques (93.28%). This is the same case for RF plus OSS, and extreme gradient boosting (XGB) plus OSS. Notably, these top ten prediction techniques significantly outperform the baseline classifier with the precision value only 0. It is concluded that random forest is the technique winning the top precision value.

According to Figure 7.10, it is clearly seen that the highest recall value (97.15%)

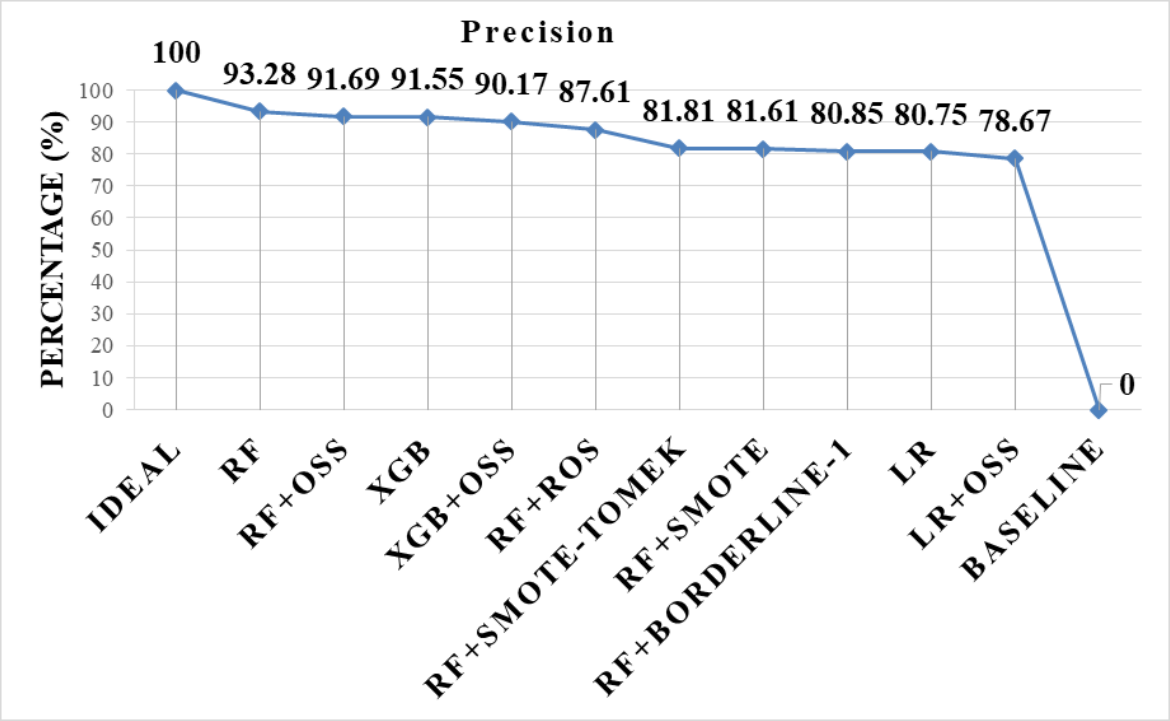


Figure 7.9: Top 10 techniques measured by precision

is achieved by extreme gradient boosting (XGB) plus NearMiss-2. It is also observed that for the predictors of XGB and random forest (RF), NearMiss-2 outperforms its counterpart NearMiss-1 regarding recall value. For instance, the recall value of NearMiss-2 plus XGB is 2.92% higher than that of NearMiss-1 plus XGB. Similarly, the recall value of NearMiss-2 plus RF is 2.86% higher than that of NearMiss-1 plus RF. Moreover, it is interesting to find that artificial neural network (ANN) occupies three positions (i.e., ANN + Boderline-2, ANN +NearMiss-1, and ANN + ADASYN) in the top 10 prediction techniques regarding recall value. Specifically, ANN plus Borderline-2 surpasses ANN plus NearMiss-1, and ANN plus NearMiss-1 performs slightly better than ANN plus ADASYN. Since the recall value of Baseline classifier is only 0, these top 10 prediction techniques considerably improve their recall value. It is concluded that extreme gradient boosting (XGB) plus NearMiss-2 is the technique winning the top recall value.

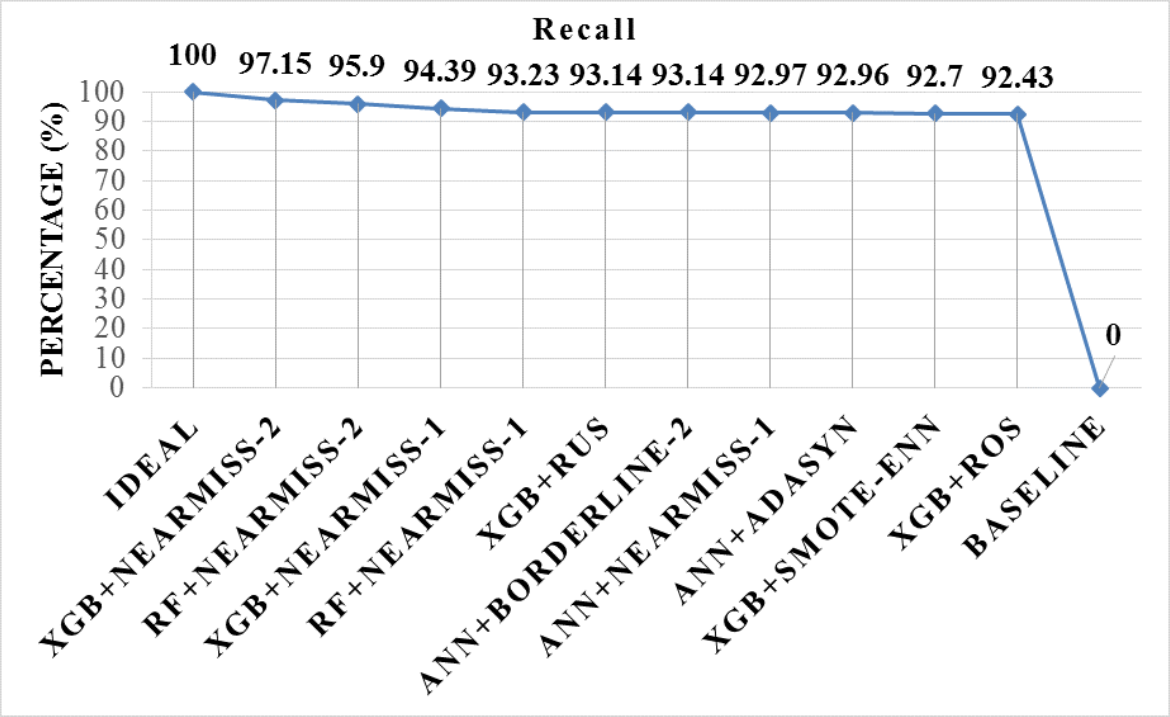


Figure 7.10: Top 10 techniques measured by recall

Figure 7.11 shows that random forest (RF) and extreme gradient boosting (XGB) play a dominant role in those techniques achieving top ten F_2 values. From the perspective of resampling techniques, the top ten F_2 values are only achieved by oversampling techniques. Especially, SMOTE, its variants the family of Borderline, two combined resampling method (SMOTE-Tomek and SMOTE-ENN), and random oversampling (ROS) are the most outstanding methods to get a high F_2 value. RF beats XGB regarding F_2 score because the top 4 F_2 values are all achieved by RF. Specifically, RF plus Borderline-2 won the top F_2 value, followed by RF plus SMOTE-ENN, RF plus SMOTE, and RF plus Borderline-1. It is concluded that random forest (RF) plus Borderline-2 is the technique winning the top F_2 value.

Figure 7.12 presents that still random forest (RF) and extreme gradient boosting (XGB) stand out in the top ten ROCs. Regarding resampling techniques, the top ten ROC values are dominantly achieved by oversampling techniques. Only one undersampling technique (random undersampling) achieved a decent ROC value. Especially,

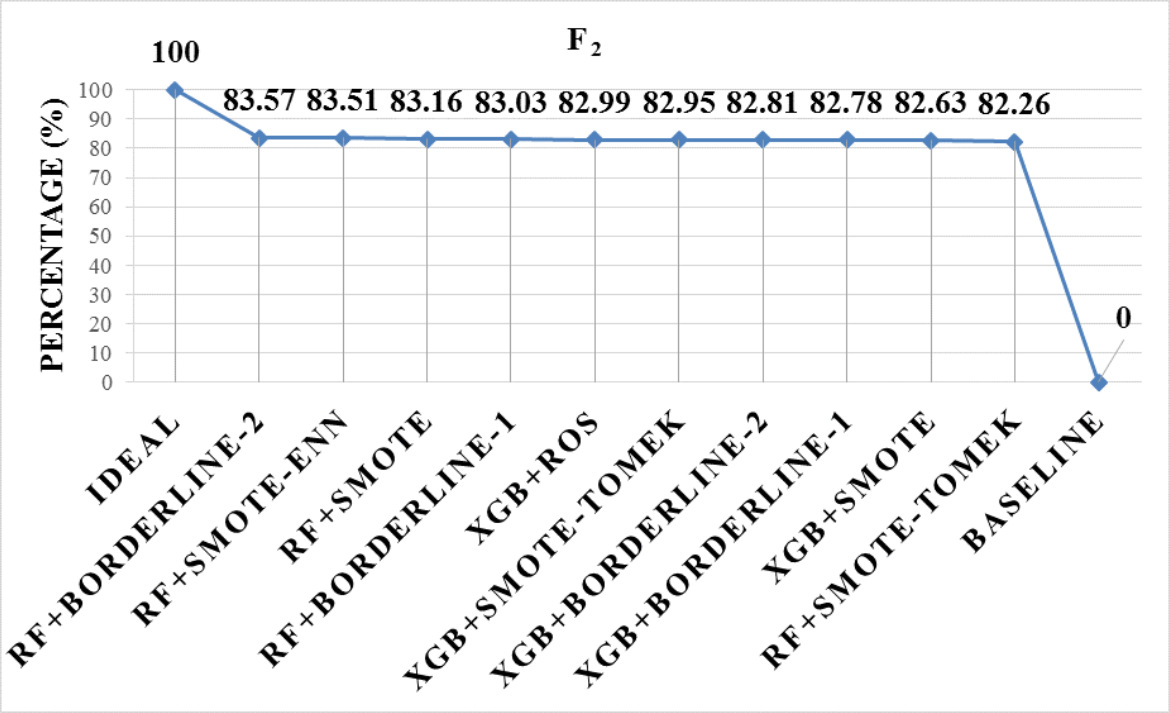


Figure 7.11: Top 10 techniques measured by F_2

random oversampling (ROS), SMOTE, its variants the family of Borderline, ADASYN, two combined resampling method (SMOTE-Tomek and SMOTE-ENN), and random undersampling are the most superior methods to get a high ROC value. This time XGB excels RF because the top 8 ROC values are all achieved by XGB. Specifically, XGB plus ROS ranks the top, followed by XGB plus Borderline-2, and XGB by SMOTE-ENN. It is concluded that extreme gradient boosting (XGB) + random oversampling (ROS) is the technique winning the top ROC value.

We summarise our finding as follows:

- The technique of winning the top accuracy and precision value: random forest (RF)
- The technique of winning the top recall value: extreme gradient boosting (XGB) + NearMiss-2
- The technique of winning the top F_2 value: random forest (RF) + Borderline-2

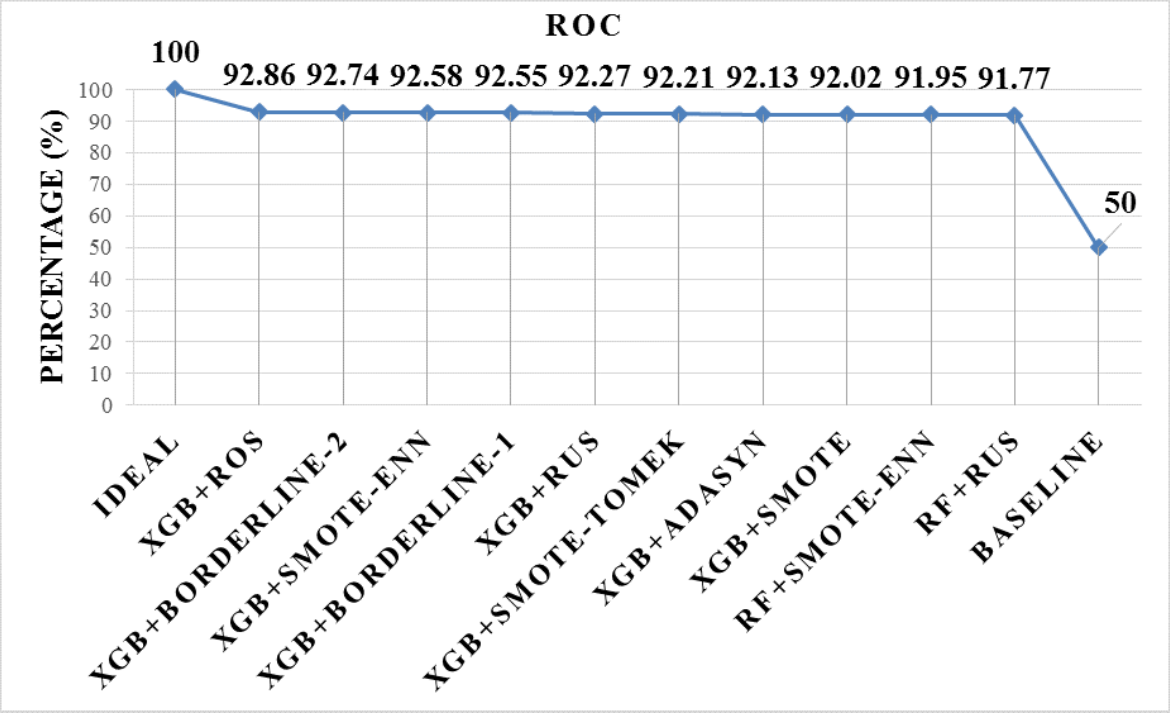


Figure 7.12: Top 10 techniques measured by ROC

- The technique of winning the top ROC value: extreme gradient boosting (XGB) + random oversampling (ROS)

Figure 7.13 shows the radar chart of the above four prediction techniques upon the five evaluation metrics.

7.11 Mathematical Model and Assumptions on Providers' Profit

So far, we identified the four prediction techniques in achieving the highest value of accuracy, precision, recall, F_2 and ROC respectively. However, it is not clear for providers how each prediction technique impacts the profit they can earn. Hence, providers would not be able to choose which prediction technique is best given their workload characteristics. In order to answer this question, we further investigate the capability of these four prediction techniques in achieving the profit and compare them

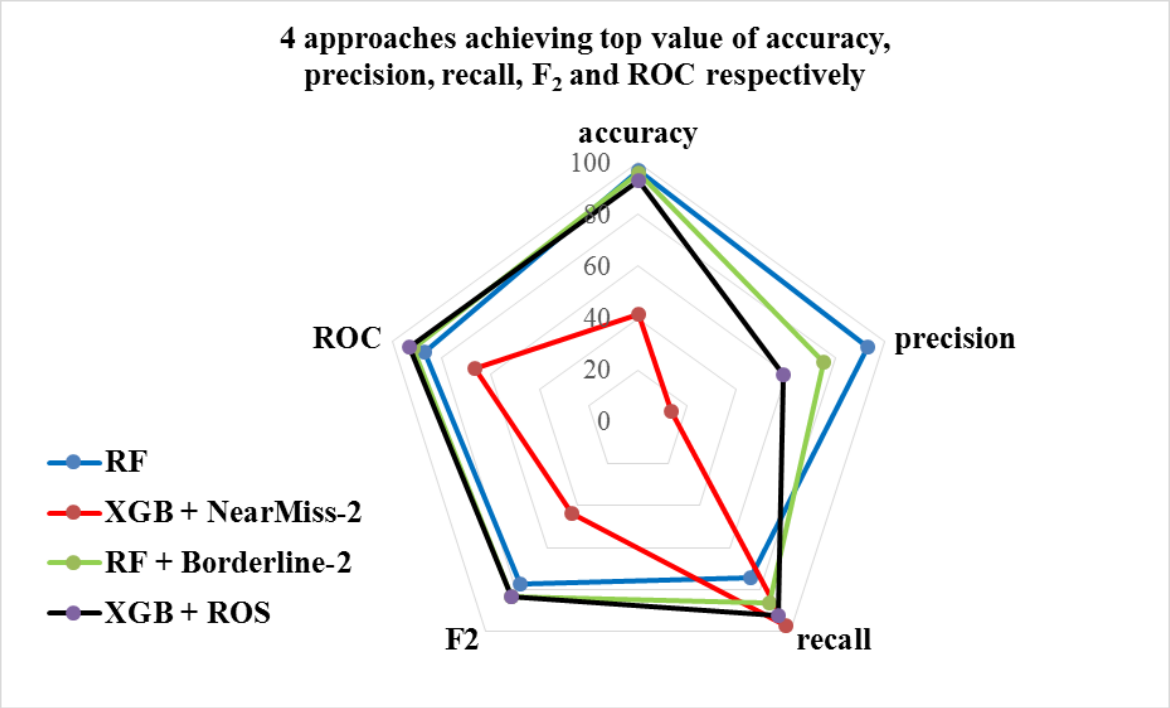


Figure 7.13: Radar chart of the four prediction techniques based on the five evaluation metrics

with the Ideal and Baseline classifiers.

Firstly, for clarity and quick reference, we develop a set of mathematical symbols in Table 7.9 to characterize the elements in our mathematical model frequently used hereafter.

For simplicity, we assume that each job J_i in J shares the equal expense E and revenue R, hence, their profit margin is also equal. Therefore, we have the following formulas:

$Margin = \frac{R-E}{R}$

$Profit_{max} = Margin * R * (N_{TN} + N_{FP})$

7.11.1 Admission Control Policy and Profit Loss Matrix

Given the outcome of each prediction technique, providers consider the following admission control policies shown in Table 7.10 in order to optimize their profit.

Table 7.9: Notation used in the mathematical model of providers' profit

Symbols	**Description**
J	A set of batch jobs or workloads
J_i	a batch job or workload instance in J
N_{TN}	The quantity of True Negative (TN) jobs in J
N_{FP}	The quantity of False Positive (FP) jobs in J
N_{FN}	The quantity of False Negative (FN) jobs in J
N_{TP}	The quantity of True Positive (TP) jobs in J
E_i	The expense of processing a job J_i
R_i	The revenue that a provider gets if successfully processing a job J_i
E_i	the expense of processing a job J_i
$Margin_i$	The profit margin that a provider makes if successfully processing a job J_i
$PenaltyRate$	The penalty ratio when a provider violated SLA
$Profit_J$	The actual profit that a provider makes after processing J
$Profit_{max}$	The maximized profit that a provider makes
$ProfitRatio$	The percentage of the actual profit after processing J over the maximized profit that a provider makes. It is defined by $\frac{Profit_J}{Profit_{max}}$

- Accept: provider will accept a batch job J_i if it is predicted as "non-violated". In this case, N_{TN} plus N_{FP} jobs will be accepted by providers.
- Reject: providers will reject a batch job J_i if it is predicted as "violated". In this case, N_{FN} plus N_{TP} batch jobs will be rejected by providers.

Table 7.10: Admission control policy matrix by providers

	Violated (Real)	**Non-Violated (Real)**
Violated (Prediction)	Reject	Reject
Non-Violated (Prediction)	Accept	Accept

According to Table 7.10, there are four situations considered:

- TN (True Negative): means a job J_i is actually "non-violated" and the prediction technique successfully predicts it as "non-violated". This is a good situation, and hence no profit loss will be produced. Clearly, providers will accept this job J_i

- FN (False Negative): means a job J_i is actually "violated" and the prediction technique misclassifies it as "non-violated". In this case, providers will accept this job J_i. However, processing J_i in the production environment will cause loss not only the cost E, but also the penalty (equals to $PenaltyRate * E$) generated due to SLA violation contract. The total profit loss can be calculated as $(1 + PenaltyRate) * E$

- TP (True Positive): means the job J_i is actually "violated" and the prediction technique successfully predicts it as "violated". This is a good situation and providers would reject J_i. In this case, no profit loss will be produced

- FP (False Positive): means the job J_i is actually "non-violated" and the classifier mistakenly predicts it as "violated". In this case, providers will reject to process this job J_i. However, giving up processing J_i in the production environment will cause the profit loss with the value of $(R - E)$.

As profit loss is generated by the either accepting or rejecting admission control policy, matrix based on the profit loss of every decision can be derived from the above admission control matrix and can be useful to evaluate the business value (profit-oriented) of a prediction technique. The details of profit loss information are shown in Table 7.11.

Table 7.11: Profit loss matrix for SLA violation detection problem (Unit: dollars/job)

	Violated (Real)	**Non-Violated (Real)**
Violated (Prediction)	0	*-(R-E)*
Non-Violated (Prediction)	*(1+PenaltyRate) * E*	0

7.11.2 The Formulation of a Provider's Profit

Now, we can derive the formula of a provider's profit $Profit_J$ after processing J based on the SLA violation detection outcome by a particular prediction technique.

$$Profit_J = (R - E) * (N_{TN} + N_{FP}) - (1 + PenaltyRate) * E * N_{FN} - (R - E) * N_{FP}$$
$$= Margin * R * N_{TN} - (1 + PenaltyRate) * R * (1 - Margin) * N_{FN}$$

Then, the $ProfitRatio$ can be calculated according to the following formula:

$$ProfitRatio = \frac{Profit_J}{Profit_{max}}$$
$$= \frac{Margin * R * N_{TN} - (1 + PenaltyRate) * R * (1 - Margin) * N_{FN}}{Margin * R * (N_{TN} + N_{FP})}$$
$$= \frac{Margin * N_{TN} - (1 + PenaltyRate) * (1 - Margin) * N_{FN}}{Margin * (N_{TN} + N_{FP})}$$

It is seen that $ProfitRatio$ is irrelevant to the revenue R, but dependents on the $PenaltyRate$ and $ProfitMargin$. In order to explore how a provider's profit varies as the $PenaltyRate$ and $ProfitMargin$ changes given by the outcome of a prediction technique, we consider three types of $PenaltyRate$ = [Low, Medium, High], which matches Alibaba's SLA definition [49], and three different types of $ProfitMargin$ = [Low, Medium, High]. Their possible combinations are detailed in Table 7.12.

Table 7.12: Different combination of $PenaltyRate$ and $ProfitMargin$

$PenaltyRate$	$PenaltyMargin$	Comment
0.1	0.1	Low $PenaltyRate$, Low $ProfitMargin$
0.25	0.1	Medium $PenaltyRate$, Low $ProfitMargin$
1	0.1	High $PenaltyRate$, Low $ProfitMargin$
0.1	0.2	Low $PenaltyRate$, Medium $ProfitMargin$
0.25	0.2	Medium $PenaltyRate$, Medium $ProfitMargin$
1	0.2	High $PenaltyRate$, Medium $ProfitMargin$
0.1	0.3	Low $PenaltyRate$, High $ProfitMargin$
0.25	0.3	Medium $PenaltyRate$, High $ProfitMargin$
1	0.3	High $PenaltyRate$, High $ProfitMargin$

7.12 Experiments and Discussion

In order to measure how much profit a prediction technique can generate based on its prediction outcome, we calculate the value of $ProfitRatio$ using the above formula for each of the four prediction techniques over the nine different combinations of $PenaltyRate$ (abbreviated as PR in Figure 7.14) and $ProfitMargin$ (abbreviated as PM in Figure 7.14) described in Table 7.12. The result in comparison with the $ProfitRatio$ value of the Ideal and Baseline classifier is shown in Figure 7.14.

It is found that for each prediction technique using the same $PenaltyRate$, the value of $ProfitRatio$ increases as the profit margin increases. This is because providers can get more profit from a prediction technique that correctly predicts more true negative jobs. For example, extreme gradient boosting (XGB) plus NearMiss-2 (the top recall winner) increases its $ProfitRatio$ by 4.98% from low profit margin to medium $ProfitMargin$, and 1.59% from medium $ProfitMargin$ to high $ProfitMargin$ given a low $PenaltyRate$.

Overall, XGB plus random oversampling (ROS) and random forest (RF) plus Borderline-2 are the top two prediction techniques to acquire profit. Next is RF (the top accuracy and precision value winner), while XGB+NearMiss-2 (the top recall value

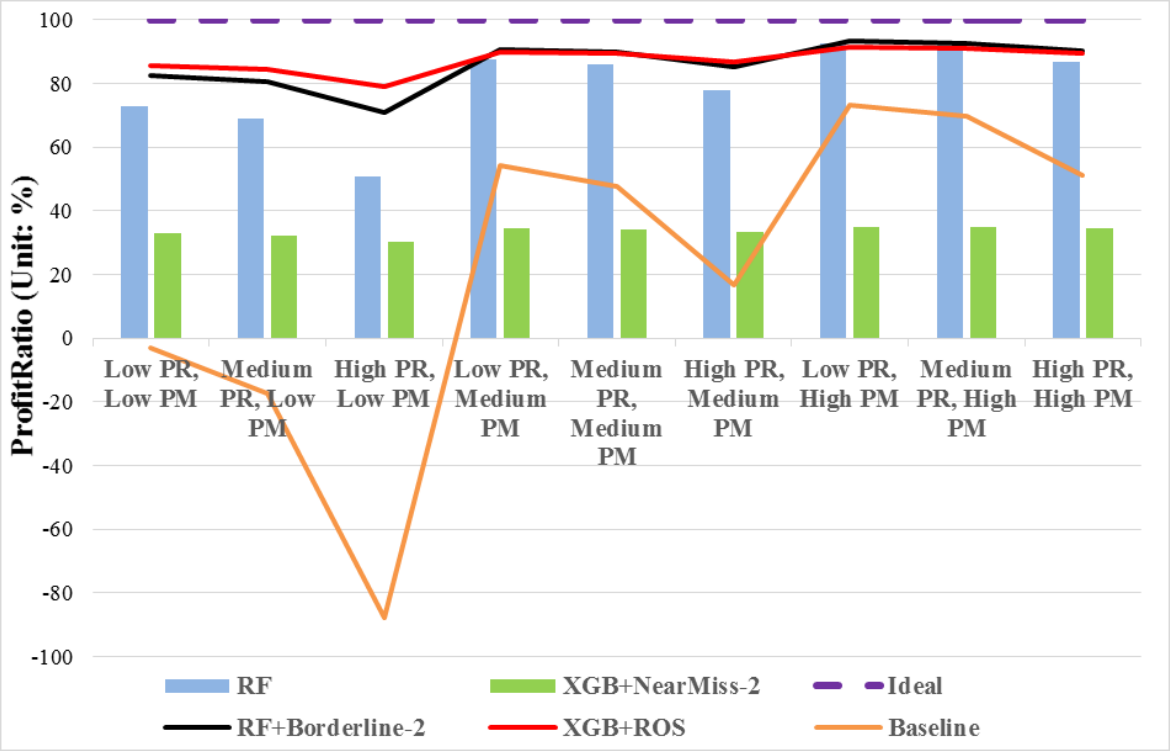


Figure 7.14: The capability of a prediction technique in achieving profit on different $PenaltyRate$ and $ProfitMargin$ combinations

winner) ranks the last. Specifically, when a low $ProfitMargin$ is applied, it is seen that XGB plus ROS (the top ROC winner) clearly outperforms RF plus borderline-2 (the top F_2 value). The most improvement ratio by XGB+ROS over RF+borderline-2 is 11.7% when a high $PenaltyRate$ is set. On the contrary, if a high $ProfitMargin$ is set, it is seen that RF plus borderline-2 (the top F_2 value) is slightly stronger than XGB plus ROS (the top ROC winner) regarding the capability of making a profit. Similarly, if a medium $ProfitMargin$ is applied, RF plus borderline-2 (the top F_2 value) is gently better than XGB plus ROS (the top ROC winner) except a high $PenaltyRate$ is set.

Even though XGB plus NearMiss-2 achieves the top recall value, it performs worse in maximizing the profit. The reason is that XGB+NearMiss-2 featured by a very high recall value (97.15%) and an extremely low precision value (only 13.64%), suffers lots of false positive cases. According to the admission control policies, lots of false positive misclassification makes providers reject these batch jobs, and hence causes a heavy loss

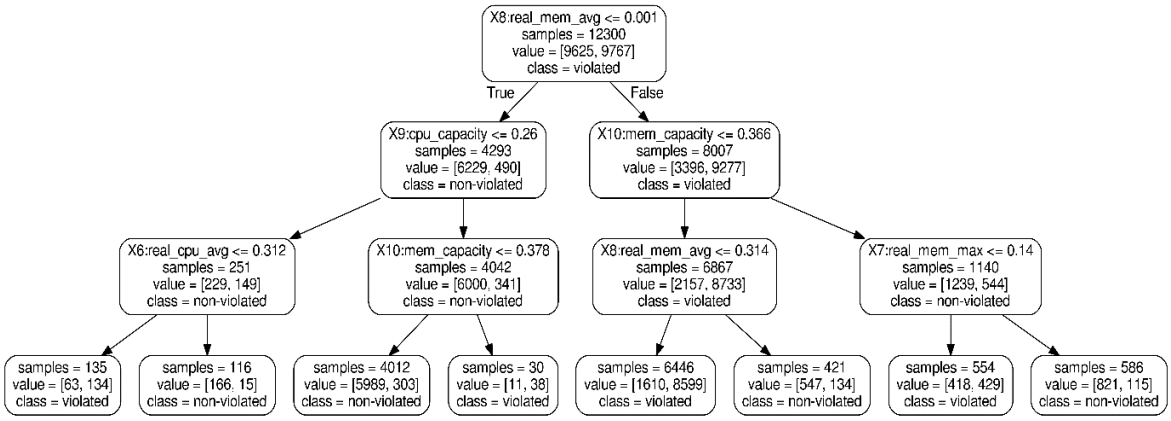
of profit. Comparatively, RF plus borderline-2 (precision = 75.36%, recall = 86.02%), XGB plus ROS (precision = 59.12%, recall = 92.43%), and RF (precision = 93.28%, recall = 74.08%) reasonably balanced the precision and recall, and hence they achieve a more decent profit.

Moreover, it is observed that the Baseline classifier fluctuates significantly over the nine combinations of $PenaltyRate$ and $ProfitMargin$. Specifically, when a low $ProfitMargin$ is applied, the profit it gets is minus. This is because the precision and recall value of Baseline classifier is zero, which causes a very high penalty, and the benefit from correctly predicting true negative jobs is very limited when a low $ProfitMargin$ is set. This situation changes when higher $ProfitMargin$ (e.g., medium $ProfitMargin$ and high $ProfitMargin$) are applied. It can be concluded that RF plus borderline-2 and XGB plus ROS are the two best prediction techniques with that providers can make optimal profit based on their prediction outcome of SLA violation.

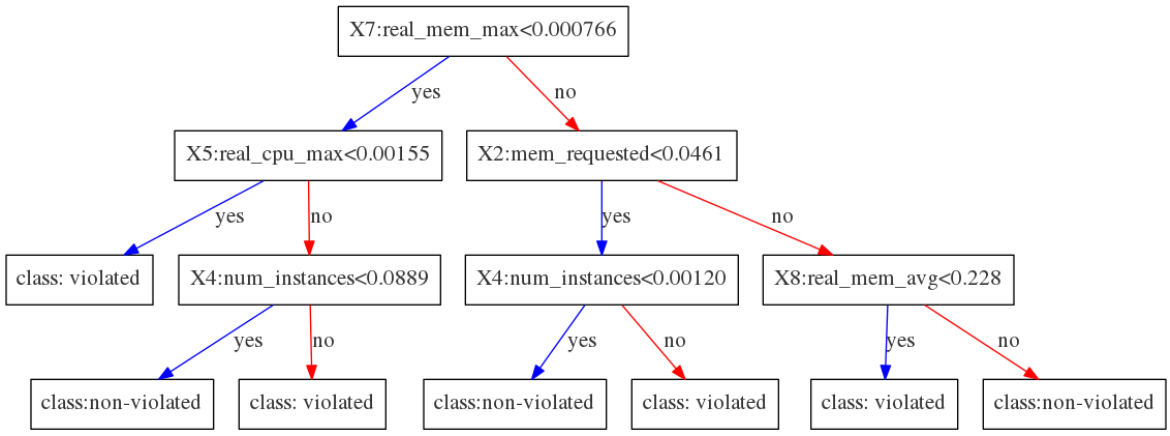
It is worth noting that both RF and XGB provide human interpretable results. According to the aforementioned working mechanism of RF and XGB, multiple single decision trees are generated and work together towards the final prediction outcome. Each single decision tree provides a very intuitive way to understand how it works on the prediction, because it follows a method of decision-making that is very similar to how humans make decisions with a chain of simple rules. Both RF and XGB can visualize any single decision trees inside them. An example of a selected single decision trees (the depth of the tree is set to 3 for better visualization) for RF and XGB respectively is shown in Figure 7.15.

It is observed from Figure 7.15(a) that among totally five appeared features, there are three features (i.e.,X_6, X_7, and X_8) belonging to BDPaaS layer. Similarly, according to Figure 7.15(b), among totally five appeared features, there are four features (i.e.,X_4, X_5, X_7, and X_8) belonging to BDPaaS layer. This demonstrates that the features at BDPaaS layer (i.e.,$X_4 - X_8$) play a dominant part in determining the final violation status compared with other features from BDSaaS and CIaaS layer.

In addition, both RF and XGB allow us to disclose each feature's importance in the classification of SLA violation. Based on our well-trained model, the average importance of these ten features based on 10-fold stratified cross validation is shown in Figure



(a) Random forest plus Borderline-2

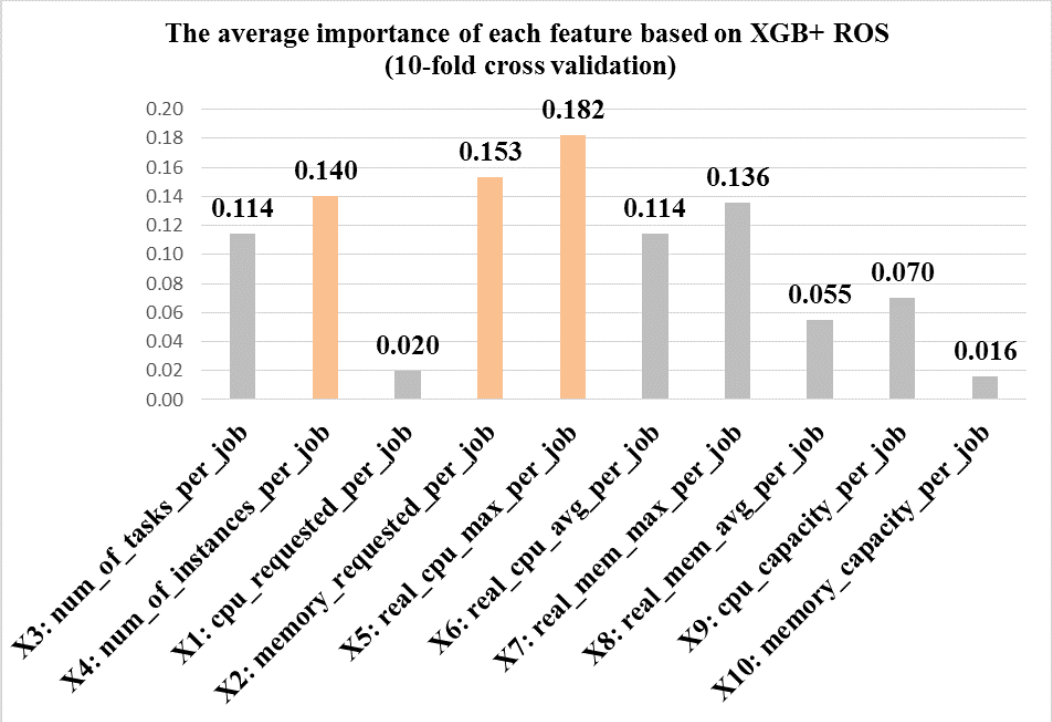


(b) Extreme gradient boosting plus random oversampling

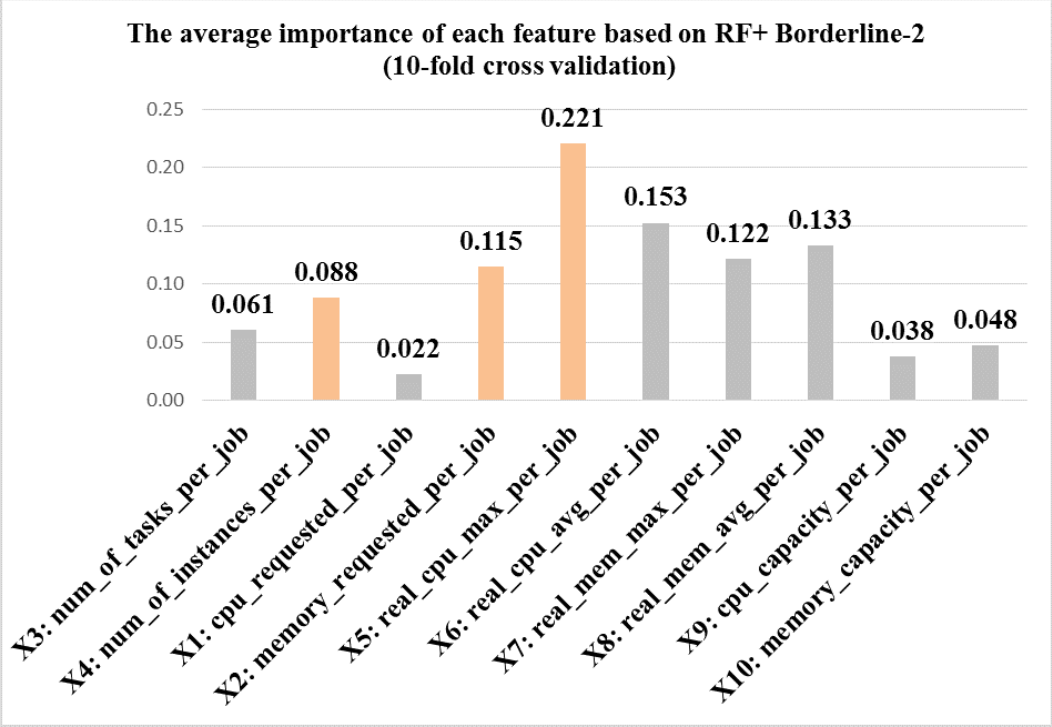
Figure 7.15: An example of selected single decision tree for RF and XGB respectively

7.16(a) for random forest (RF) plus Borderline-2 and in Figure 7.16(b) for extreme gradient boosting (XGB) plus random oversampling (ROS) respectively. It is observed that the *real_cpu_max* has the highest contribution, followed by *memory_requested* and the number of instances for XGB plus ROS. Also, it is seen that the *real_cpu_max* has the top contribution, and the next is *real_cpu_avg*, followed by *real_mem_avg* for RF + Borderline-2. Moreover, *memory_capacity* and *cpu_requested* have much lower contributions than other features in both RF and XGB predictors.

Further, we aggregate the feature importance by layers and present the graphical representation in Figure 7.17. It is found that for XGB plus ROS, the aggregated features importance at BDPaaS layer is dominant, occupying 63%, roughly two times



(a) Random forest plus Borderline-2



(b) Extreme gradient boosting plus random oversampling

Figure 7.16: The average importance of each feature in RF and XGB respectively

of that at BDSaaS layer (29%), and eight times of that at CIaaS layer (8%). Similarly, for RF plus Borderline-2, the aggregated features importance at BDPaaS layer is also dominant, occupying 72%, roughly three times of that at BDSaaS layer (20%), and nine times of that at CIaaS layer (8%).

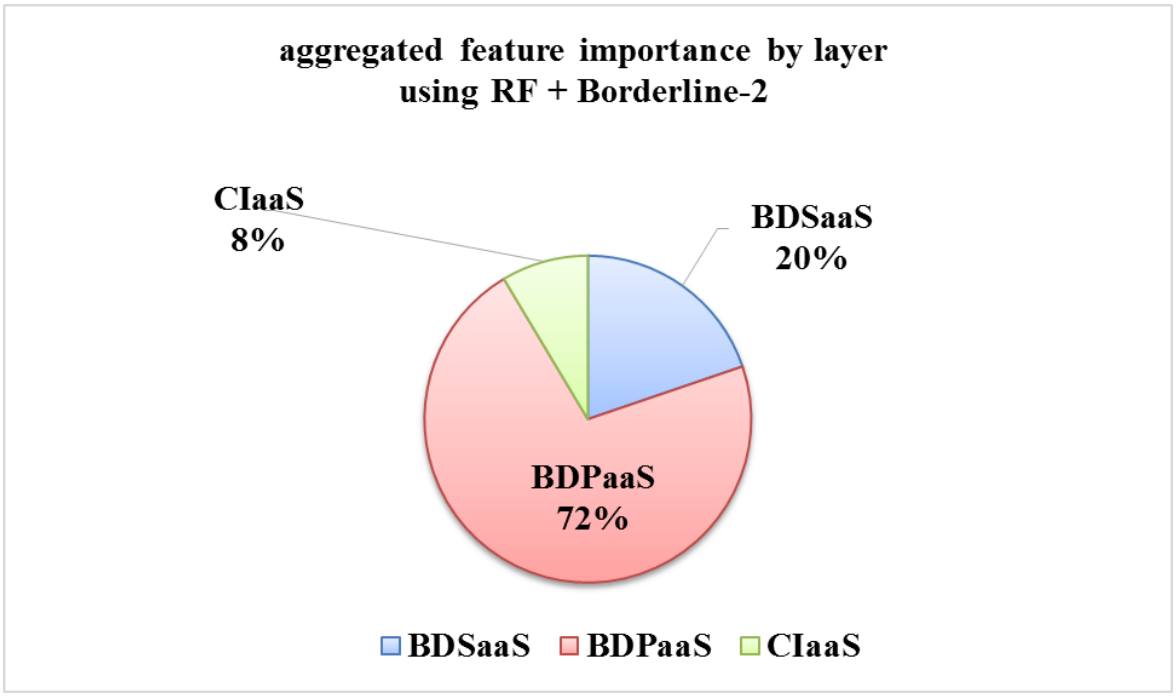
It can be concluded that both XGB and RF attach more importance to the features from BDPaaS layer (i.e., $real_cpu_max$, $real_cpu_avg$, $number_of_instances$, and $real_mem_avg$) compared with the feature ($cpu_requested$) at BDSaaS layer and the feature ($memory_capcacity$) at CIaaS layer. This is a further evidence that the features at BDPaaS layer (i.e., $X_4 - X_8$) are determinant factors to detect final violation status.

Such findings uncover the hidden patterns of the multiple configurations across layers and provide insightful information to providers for decision making. Concretely, BDPaaS layer needs more attention so that it can better serve the batch job workloads. For example, at BDPaaS layer, scheduling a batch job that better allocates the CPU or memory or reasonably split the job into the number of instances can improve the efficacy in reducing the SLA violations. On the contrary, paying attention to the capacity of memory or CPU at CIaaS layer, or the requested CPU at BDSaaS layer will generate insignificant efficacy in reducing the SLA violations.

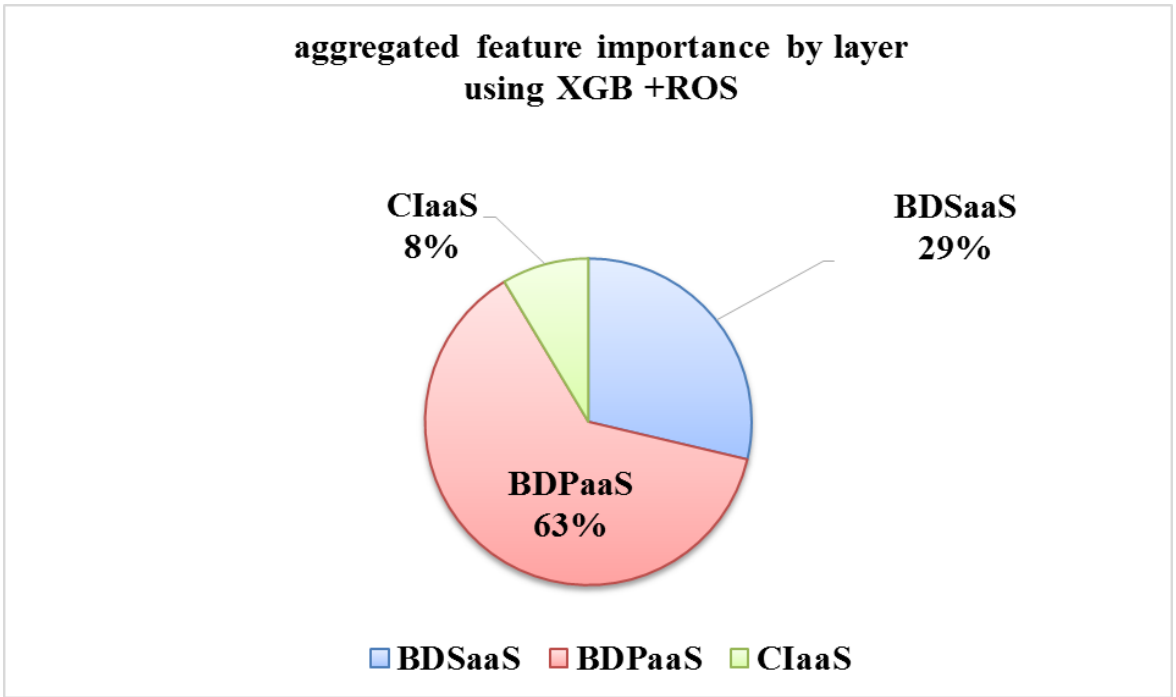
It is worthy mentioning that (i) our proposed SLA violation detection techniques are readily generalized to the wide context of machine learning / artificial intelligence applications (e.g., neural network, regressions, classifications etc.,) as long as the features of such applications can be captured; (ii) our proposed SLA violation detection techniques can also be applied to near-real time scenario to detect violations before they happen. That is to say, for any incoming application in a near-real time scenario, we can predict its SLA violation outcome timely based on our well-trained and saved prediction models.

7.13 Summary

We addressed the problem of detecting SLA violations for a real cloud-hosted BDAA in this Chapter. We used the dataset that is newly released by Alibaba and



(a) Random forest plus Borderline-2



(b) Extreme gradient boosting plus random oversampling

Figure 7.17: The aggregated feature importance by layer in RF and XGB

contains detailed trace information regarding batch workloads among 1300 machines in 12 hours. We explored four diverse machine learning-based predictors (i.e., logistics regression, artificial neural network, random forest, and extreme gradient boosting) to detect the SLA violations. Since the dataset is heavily skewed, we also examined 12 different resampling techniques to handle the challenge of data skewness in order to acquire better performance. Standard statistical significance metrics, such as accuracy, precision, recall, F_2, and ROC has been applied to test the practicality and efficiency of the predictors. Most importantly, we designed a novel mathematical model regarding provider's profit and evaluated the capability of these prediction techniques in helping providers' optimizing their profits. Importantly, the hidden patterns of the multiple configurations across layers are uncovered, and insightful information is provided for providers' decision making.

Chapter 8

Conclusion and Future Works

8.1 Summary of the Research

In this thesis, we focused on the research problem: How to manage service level agreements (SLAs) for big data analytics application (BDAAs) in cloud in ensuring SLAs guarantee? This research problem is broken down into five research questions. From Chapter 3 to Chapter 7, novel approaches to address these five research questions are discussed.

Chapter 3 addressed the first research question *RQ1*: What is the landscape of the extant research on service level agreements (SLAs) management for big data analytics application (BDAAs) in cloud and how to classify them from different perspectives? In this Chapter, using a systematic literature review methodology, we conducted an in-depth literature review on the SLA management for BDAAs in cloud. Based on this comprehensive review, we proposed a novel thematic taxonomy that consists of six core dimensions including Actors, Layers, Techniques, Cloud deployment models, SLA metrics and Conceptualization. Through the lens of this review, the findings of each dimension have been discussed.

Chapter 4 addressed the research question *RQ2*: How to describe and model service level agreements (SLAs) for cloud-hosted big data analytics application (BDAAs) in a unified and structured way, and specify common and niche SLA metrics while respecting characteristics of BDAAs? We proposed a novel conceptual SLA model that is suitable for BDAAs in cloud. This model can not only demonstrate different types of SLAs (aSLA,

pSLA, and cSLA) in a structured format and provide a universal way to look at SLAs for BDAAs but also indicate their strong dependencies relationship for future researchers to understand. Also, we presented a new multi-dimensional categorization scheme regarding SLA metrics for BDAAs in cloud. This categorization scheme consists of common metrics and niche metrics for each layer and respects characteristics of BDAAs. It provides a guideline for providers and customers to understand SLA conventions between them and make design decision in particular concerning the definition and specification of SLAs.

Chapter 5 addressed the research question *RQ3*: How to model and simulate cloud-hosted big data analytics application (BDAAs) across layers to facilitate SLAs management? To demonstrate the research gap regarding the existing simulation tools and motivate a new simulator, a detailed literature review were conducted. Then, we designed and implemented a new discrete event-based simulator called IoTSim on top of Cloudsim. The layer-based architecture of IoTSim, the fundamental entities and their correlation, and implementation details have been discussed. The evaluation has proven the efficacy of IoTSim.

Chapter 6 addressed the research question *RQ4*: How to achieve SLA-driven scheduling for cloud-hosted big data analytics application (BDAAs) across layers while guaranteeing SLAs? In this chapter, we presented a high-level system model for SLA-driven scheduling BDAAs in public cloud. Next, we formulated the SLA-driven scheduling problem as a multi-objective optimization problem. We then developed and evaluated a novel SLA-driven scheduling algorithm that solves the above optimization problem. Extensive experiments not only demonstrate that the proposed algorithm outperforms the baseline NoSLA algorithm but also proves that its time-complexity is very similar to the well-known exhaustive search heuristics.

Chapter 7 addressed the research question of *RQ5*: How to detect SLA violations for cloud-hosted big data analytics application (BDAAs) across layers before they happen to maximize providers' profit? In this Chapter, we investigated the problem of detecting SLA violations in a real dataset. This dataset is released on September 2017 by Alibaba and keeps detailed trace information regarding batch workloads among 1300 machines in 12 hours, which provides a good representation of big data application deployed in cloud

computing environment. We explored four diverse machine learning-based predictors (i.e., logistics regression, artificial neural network, random forest, and extreme gradient boosting) to detect the SLA violations. Since the dataset is heavily skewed, we also examined 12 different resampling techniques to handle the challenge of data skewness in order to acquire better performance. Standard statistical significance measures, such as accuracy, precision, recall and so on has been applied to test the practicality and efficiency of the predictors. Most importantly, a mathematical formulation to model provider's profit has been built and tested the capability of these prediction techniques in helping providers' making profits.

8.2 Major Research Contributions

To the best of our knowledge, this thesis is the very earliest attempt towards the SLA management for cloud-hosted BDAAs. In summary, this thesis makes five major research contributions as following.

8.2.1 A Novel Thematic Taxonomy of SLA Management for BDAAs in Cloud Using Systematic Literature Review Methodology

We have conducted a systematic literature review of the SLA management for cloud-hosted BDAAs in Chapter 3, which leads to the first contribution (a systematic literature review contribution). To the best of our knowledge, the in-depth survey of SLA management for cloud-hosted BDAAs is very scarce. This contribution has not only provide future researchers an outlook on SLA management for the applications, but it also provides insight into understanding the different research perspectives (Actors, Layers, Techniques, Cloud deployment models, SLA metrics and Conceptualization) of many works in this research area.

8.2.2 A New SLA Model and General Categorization Scheme of SLA Metrics for BDAAs in Cloud

This second contribution (a conceptual contribution) is presented in Chapter 4. The work is one of the first attempts comprehensively investigating the SLA management problem in the context of cloud-hosted big data analytics applications. Current literature lacks the depth as well as the breadth of knowledge required to model and understand SLAs for BDAAs. The proposed SLA model not only facilitates future researchers to understand SLAs for cloud-hosted BDAAs in a universal and structured way but also offers a novel categorization scheme of SLA metrics. Thus the proposed model establishes a perfect guideline for providers and customers to understand SLA conventions between them and make decision in particular concerning the definition and specification of SLAs in the context of cloud-hosted BDAAs.

8.2.3 A New Simulation Toolkit for Modeling and Simulating BDAAs in Cloud

We have designed and developed a new simulator in Chapter 5, which leads to the third contribution (a technical contribution). This simulator is built on top of Cloudsim and extensively extends and enhances its functionality and capabilities by effectively modeling and simulating MapReduce-based big data analytics applications in cloud. Future readers can successfully model and simulate the features, components, and behaviors of MapReduce as it does in the real world. Also, this simulator offers an efficient tool for future readers to perform experiments under non-static conditions (e.g., availability and workload pattern) in a controllable environment where tests can be re-executed, evaluated the performance or efficacy of their solutions if meeting SLA requirements.

8.2.4 A New Cost-efficient SLA-driven Scheduling Algorithm for MapReduce applications

A new cost-efficient SLA-driven scheduling algorithm for MapReduce-based applications has been designed and implemented in Chapter 6. This is the fourth contribution

(a technical contribution). This novel scheduling algorithm not only satisfies SLA requirements agreed between providers and customers but also minimizes the cost for customers while provisioning an optimal allocation of the public cloud resources to Hadoop (an open source implementation of MapReduce programming model) processes. The extensive experiments clearly show that our proposed SLA-driven algorithm can help users reduce the cost of executing MapReduce-based BDAAs on public clouds by about 25% to 50% while meeting diverse SLA requirements. This work successfully delivers a robust SLA-driven algorithm to optimize Hadoop performance on public clouds given multiple SLA constraints.

8.2.5 A Novel SLA Violation Detection for BDAAs in Cloud

We successfully detect SLA violations for Alibaba's batch workloads deployed in cloud environment through the exploration and evaluation of the four representative machine learning techniques (i.e., Logistics Regression, Artificial Neural Network, Random Forest, and Extreme Gradient Boosting) with 12 diverse resampling techniques (5 oversampling, 5 undersampling and 2 combined resampling), which leads to the fifth contribution (a technical contribution). Moreover, we design a mathematical model to formulate the provider's profit and investigate how the outcome of a prediction technique impact providers' profit. This work is one of the first attempts towards detection of SLA violations for big data analytics applications using machine learning and data skewness handling techniques. This work will help providers to choose the best performing prediction technique, and most importantly, it uncovers the hidden patterns of the multiple configurations across layers and provides insightful information for providers' decision making.

8.3 Limitations and Future Works

The work in Chapter 5 regarding the proposed simulators is only suitable for simulating and modeling performance of the MapReduce programming model in cloud environment. Hence, it is insufficient in the context of modeling and simulating the behavior of cloud-hosted stream processing model. Since stream processing has recently

received increasing attention, future work could study on the modeling and simulation of SLA management for cloud-hosted stream processing model. Our work will be beneficial in doing this new SLA work in the context of stream model because the proposed architecture in Section 5.5 provides an extensible and scalable framework for future researchers who can develop new functions and capabilities on top of it to support modeling and simulating stream-based application workloads.

The work in Chapter 6 regarding the proposed SLA-driven scheduling algorithm is limited to optimal scheduling MapReduce-based BDAAs. It is not able to schedule stream-based applications yet. Future work could study how to achieve SLA-driven scheduling for these stream-based applications. Our work is constructive in addressing the challenge of SLA-driven scheduling in the context of stream processing model because the proposed algorithms and evaluation methodology set good examples for future researchers to formulate a new mathematical model and develop new algorithm on top of it in achieving the optimal schedule of stream-based applications.

Two future works might arise from our studies in Chapter 7 regarding the detection of SLA violation for BDAAs in cloud. On the one hand, we formulated the SLA violation detection problem as a binary classification problem. Future work could apply anomaly detection-based technique to identify unusual patterns that do not conform to expected behavior (i.e., the batch job is "violated") in the Alibaba dataset. Our work is very consutructive for future researchers in the application of the anomaly detection technique to detect SLA violations for cloud-hosted BDAAs because the extracted features set and proposed mathematical profit model can be utilized by future researchers. On the other hand, we only explored the SLA violation detection for batch-based workload. Future work could extend our prediction techniques and data skewness handling techniques to steam-based BDAA workload. Still, our work is very helpful for future researchers to conduct this investigation.

Last but not least, the advent of Fog Computing and Edge Computing (See Section 2.6.2.2) also bring future works towards SLA-specific management in such context. These two new computing paradigms as powerful complimentary architecture to Cloud Computing have attracted research attentions recently and also have a strong need of SLA to manage the level and quality of service [178]. Although the results of our thesis

are specifically targeted to the SLA management for BDAAs in Cloud environment, they undoubtedly provide lots of insight on how to manage SLA for BDAAs in the context of Fog Computing and Edge Computing. Our proposed SLA model, categorization scheme of SLA metrics, and SLA violation detection techniques could be readily extended and applied to such context for future researchers.

It is believed that the above (but not limited to) future research directions will advance the technology presented in this thesis and contribute to academia and industry.

Bibliography

[1] "The World Economic Forum:The Fourth Industrial Revolution: what it means, how to respond." https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/.

[2] Yang Lu. "Industry 4.0: A survey on technologies, applications and open research issues". In: *Journal of Industrial Information Integration* 6 (2017), pp. 1–10.

[3] Vasja Roblek, Maja Meško, and Alojz Krapež. "A complex view of industry 4.0". In: *Sage Open* 6.2 (2016), p. 2158244016653987.

[4] Saint John Walker. "Big data: A revolution that will transform how we live, work, and think". 2014.

[5] Michael Rüßmann et al. "Industry 4.0: The future of productivity and growth in manufacturing industries". In: *Boston Consulting Group* 9.1 (2015), pp. 54–89.

[6] Jay Lee, Hung-An Kao, and Shanhu Yang. "Service innovation and smart analytics for industry 4.0 and big data environment". In: *Procedia Cirp* 16 (2014), pp. 3–8.

[7] Rainer Schmidt et al. "Industry 4.0-potentials for creating smart products: empirical research results". In: *International Conference on Business Information Systems*. Springer. 2015, pp. 16–27.

[8] Seref Sagiroglu and Duygu Sinanc. "Big data: A review". In: *2013 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE. 2013, pp. 42–47.

[9] Shen Yin and Okyay Kaynak. "Big data for modern industry: challenges and trends [point of view]". In: *Proceedings of the IEEE* 103.2 (2015), pp. 143–146.

[10] "Big data is the fourth industrial revolution." https://www.manufacturingglobal.com/technology/big-data-fourth-industrial-revolution. Nov,2016.

[11] "IBM: What is big data? - Bringing big data to the enterprise." https://www.ibm.com/analytics/hadoop/big-data-analytics.

[12] John Gantz and David Reinsel. "The 2011 IDC digital universe study: Extracting value from chaos". June 2011.

[13] Martin Hilbert and Priscila López. "The world's technological capacity to store, communicate, and compute information". In: *science* 332.6025 (2011), pp. 60–65.

[14] Feng Xia et al. "Internet of things". In: *International Journal of Communication Systems* 25.9 (2012), p. 1101.

[15] Yunchuan Sun et al. "Internet of things and big data analytics for smart and connected communities". In: *IEEE access* 4 (2016), pp. 766–773.

[16] Lizhe Wang and Rajiv Ranjan. "Processing distributed internet of things data in clouds". In: *IEEE Cloud Computing* 2.1 (2015), pp. 76–80.

[17] John Gantz and David Reinsel. "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east". In: *IDC iView: IDC Analyze the future* 2007.2012 (2012), pp. 1–16.

[18] Ala Al-Fuqaha et al. "Internet of things: A survey on enabling technologies, protocols, and applications". In: *IEEE communications surveys & tutorials* 17.4 (2015), pp. 2347–2376.

[19] IDC. "Worldwide Internet of Things (IoT) 2013-2020 Forecast: Billions of Things, Trillions of Dollars". Doc #243661, October 2013.

[20] Charith Perera et al. "Big data privacy in the internet of things era". In: *IT Professional* 17.3 (2015), pp. 32–39.

[21] Charith Perera et al. "Context aware computing for the internet of things: A survey". In: *IEEE communications surveys & tutorials* 16.1 (2013), pp. 414–454.

[22] Chaowei Yang et al. "Big Data and cloud computing: innovation opportunities and challenges". In: *International Journal of Digital Earth* 10.1 (2017), pp. 13–53.

[23] Mohsen Marjani et al. “Big IoT data analytics: architecture, opportunities, and open research challenges”. In: *IEEE Access* 5 (2017), pp. 5247–5261.

[24] Uthayasankar Sivarajah et al. “Critical analysis of Big Data challenges and analytical methods”. In: *Journal of Business Research* 70 (2017), pp. 263–286.

[25] Rajiv Ranjan et al. “Cloud computing: methodology, systems, and applications”. CRC Press, 2011.

[26] Anthony D JoSEP et al. “A view of cloud computing”. In: *Communications of the ACM* 53.4 (2010).

[27] Armando Fox et al. “Above the clouds: A berkeley view of cloud computing”. In: *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS* 28.13 (2009), p. 2009.

[28] Philipp Gölzer, Patrick Cato, and Michael Amberg. “Data Processing Requirements of Industry 4.0-Use Cases for Big Data Applications.” In: *ECIS*. 2015.

[29] Claudio A Ardagna, Paolo Ceravolo, and Ernesto Damiani. “Big data analytics as-a-service: Issues and challenges”. In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 3638–3644.

[30] Farhana Zulkernine et al. “Towards cloud-based analytics-as-a-service (cIaaas) for big data analytics in the cloud”. In: *2013 IEEE International Congress on Big Data*. IEEE. 2013, pp. 62–69.

[31] Zhaohao Sun, Huasheng Zou, and Kenneth Strang. “Big data analytics as a service for business intelligence”. In: *Conference on e-Business, e-Services and e-Society*. Springer. 2015, pp. 200–211.

[32] Marco Anisetti et al. “Privacy-aware Big Data Analytics as a service for public health policies in smart cities”. In: *Sustainable cities and society* 39 (2018), pp. 68–77.

[33] Emmanuel Marilly et al. “Requirements for service level agreement management”. In: *IEEE Workshop on IP Operations and Management*. IEEE. 2002, pp. 57–62.

[34] Damián Serrano et al. "SLA guarantees for cloud services". In: *Future Generation Computer Systems* 54 (2016), pp. 233–246.

[35] Dimosthenis Kyriazis. "Cloud computing service level agreements–exploitation of research results." http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=2496.

[36] Marcos D Assunção et al. "Big Data computing and clouds: Trends and future directions". In: *Journal of Parallel and Distributed Computing* 79 (2015), pp. 3–15.

[37] "GOOGLE BigQuery." https://cloud.google.com/bigquery/.

[38] "Amazon Redshift." https://aws.amazon.com/redshift/.

[39] "Microsoft Data Lake Analytics." https://azure.microsoft.com/en-us/services/data-lake-analytics/.

[40] "Google Prediction API and Google BigQuery SLA." https://cloud.google.com/bigquery/sla/.

[41] "Google BigQuery TITSUP caused by failure to scale-yer workloads." https://www.theregister.co.uk/2016/11/14/google_bigquery_outage/.

[42] "Google BigQuery Incident #18022." https://status.cloud.google.com/incident/bigquery/18022.

[43] "Google Cloud Dataflow Service Level Agreement (SLA)." https://cloud.google.com/dataflow/sla.

[44] "Google Cloud Dataproc Service Level Agreement (SLA)." https://cloud.google.com/dataproc/sla.

[45] "SLA for HDInsigh." https://azure.microsoft.com/en-us/support/legal/sla/hdinsight/v1_0/.

[46] "Amazon EMR Service Level Agreement." https://aws.amazon.com/emr/sla/.

[47] "Rackspace Hadoop Service SLA." https://www.rackspace.com/sites/default/files/legal/cloud_slas_global_08_feb_2016.pdf.

[48] "SLA for SAP Cloud Services." https://assets.cdn.sap.com/agreements/product-use-and-support-terms/cls/id/service-level-agreement-for-sap-cloud-services-englishindonesian-v4-2018.pdf.

[49] "Alibaba Elastic Compute Service Service Level Agreement." https://www.alibabacloud.com/help/doc-detail/42436.htm.

[50] "SLA for Data Lake Analytic." https://azure.microsoft.com/en-us/support/legal/sla/data-lake-analytics/v1_0/.

[51] K Saravanan and M Rajaram. "An Exploratory Study of Cloud Service Level Agreements-State of the Art Review." In: *KSII Transactions on Internet & Information Systems* 9.3 (2015).

[52] Gary Alan Fine. "Handbook of Qualitative Research." 1995.

[53] Anne Håkansson. "Portal of research methods and methodologies for research projects and degree projects". In: *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*. CSREA Press USA. 2013, pp. 67–73.

[54] R Burke Johnson, Anthony J Onwuegbuzie, and Lisa A Turner. "Toward a definition of mixed methods research". In: *Journal of mixed methods research* 1.2 (2007), pp. 112–133.

[55] Ehtiram Raza Khan and Huma Anwar. "Research Methods of Computer Science". University Science Press, 2015.

[56] Andrea De Mauro, Marco Greco, and Michele Grimaldi. "What is big data? A consensual definition and a review of key research topics". In: *AIP conference proceedings*. Vol. 1644. 1. AIP. 2015, pp. 97–104.

[57] Bart van der Sloot, Dennis Broeders, and Erik Schrijvers. "Exploring the boundaries of Big Data". Amsterdam University Press Amsterdam, 2016.

[58] Jonathan Stuart Ward and Adam Barker. "Undefined by data: a survey of big data definitions". In: *arXiv preprint arXiv:1309.5821* (2013).

[59] Doug Laney. "3D Data Management: Controlling Data Volume, Velocity, and Variety." http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf. 2001.

[60] Mark Beyer and Douglas Laney. "The Importance of Big Data: A Definition." https://www.gartner.com/en/documents/2057415/the-importance-of-big-data-a-definition. June 2012.

[61] "Big Data at NIST." http://bigdatawg.nist.gov/home.php.

[62] "The Big Data Conundrum: How to Define It." https://www.technologyreview.com/s/519851/the-big-data-conundrum-how-to-define-it/. MIT Technology Review. October 3, 2013.

[63] Amir Gandomi and Murtaza Haider. "Beyond the hype: Big data concepts, methods, and analytics". In: *International journal of information management* 35.2 (2015), pp. 137–144.

[64] Geng Lin et al. "Cloud computing: IT as a service". In: *IT professional* 2 (2009), pp. 10–13.

[65] Min Chen, Shiwen Mao, and Yunhao Liu. "Big data: A survey". In: *Mobile networks and applications* 19.2 (2014), pp. 171–209.

[66] Jeffrey Voas and Jia Zhang. "Cloud computing: New wine or just a new bottle?" In: *IT professional* 11.2 (2009), pp. 15–17.

[67] Luis M Vaquero et al. "A break in the clouds: towards a cloud definition". In: *ACM SIGCOMM Computer Communication Review* 39.1 (2008), pp. 50–55.

[68] Rajkumar Buyya et al. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". In: *Future Generation computer systems* 25.6 (2009), pp. 599–616.

[69] Peter Mell, Tim Grance, et al. "The NIST definition of cloud computing". 2011.

[70] Eugene Gorelik. "Cloud computing models". PhD thesis. Massachusetts Institute of Technology, 2013.

[71] Ahmed Shawish and Maria Salama. "Cloud computing: paradigms and technologies". In: *Inter-cooperative collective intelligence: Techniques and applications*. Springer, 2014, pp. 39–67.

[72] Peter Mell, Tim Grance, et al. "Deploying big data analytics applications to the cloud". May 2014.

[73] Ibrahim Abaker Targio Hashem et al. "The rise of "big data" on cloud computing: Review and open research issues". In: *Information systems* 47 (2015), pp. 98–115.

[74] CL Philip Chen and Chun-Yang Zhang. "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data". In: *Information sciences* 275 (2014), pp. 314–347.

[75] Ahmed Oussous et al. "Big Data technologies: A survey". In: *Journal of King Saud University-Computer and Information Sciences* 30.4 (2018), pp. 431–448.

[76] Han Ping Fung. "Using Big Data Analytics in Information Technology (IT) Service Delivery". In: *Internet Technologies and Applications Research, ITAR* 1.1 (2013), pp. 6–10.

[77] Wendy Arianne Günther et al. "Debating big data: A literature review on realizing value from big data". In: *The Journal of Strategic Information Systems* 26.3 (2017), pp. 191–209.

[78] Steve LaValle et al. "Big data, analytics and the path from insights to value". In: *MIT sloan management review* 52.2 (2011), pp. 21–32.

[79] Mininath R Bendre and Vijaya R Thool. "Analytics, challenges and applications in big data environment: a survey". In: *Journal of Management Analytics* 3.3 (2016), pp. 206–239.

[80] Hsinchun Chen, Roger HL Chiang, and Veda C Storey. "Business intelligence and analytics: From big data to big impact." In: *MIS quarterly* 36.4 (2012).

[81] Nishita Mehta and Anil Pandit. "Concurrence of big data analytics and healthcare: A systematic review". In: *International journal of medical informatics* 114 (2018), pp. 57–65.

[82] "Building Real-Time Inventory Systems for Retail." https://cloud.google.com/solutions/building-real-time-inventory-systems-retail/.

[83] "Amazon Comprehend:Discover insights and relationships in text." https://aws.amazon.com/comprehend/.

[84] Brian Hopkins. " Move Big Data To The Public Cloud With An Insight PaaS." https://go.forrester.com/blogs/insight-paas-accelerate-big-data-cloud/. August 8, 2017.

[85] " AtScale." http://atscale.com/.

[86] Georgios Skourletopoulos et al. "Big data and cloud computing: a survey of the state-of-the-art and research challenges". In: *Advances in mobile cloud computing and big data in the 5G Era*. Springer, 2017, pp. 23–41.

[87] Xi Sun et al. "A cost-effective approach to delivering analytics as a service". In: *2012 IEEE 19th International Conference on Web Services*. IEEE. 2012, pp. 512–519.

[88] Denis Lehmann, David Fekete, and Gottfried Vossen. "Technology selection for big data and analytical applications". Tech. rep. Working Papers, ERCIS-European Research Center for Information Systems, 2016.

[89] Zibin Zheng, Jieming Zhu, and Michael R Lyu. "Service-generated big data and big data-as-a-service: an overview". In: *2013 IEEE international congress on Big Data*. IEEE. 2013, pp. 403–410.

[90] Han Hu et al. "Toward scalable systems for big data analytics: A technology tutorial". In: *IEEE access* 2 (2014), pp. 652–687.

[91] Linh Manh Pham. "A Big Data Analytics Framework for IoT Applications in the Cloud". In: *VNU Journal of Science: Computer Science and Communication Engineering* 31.2 (2015).

[92] "Amazon EC2." https://aws.amazon.com/ec2/.

[93] " Amazon S3." https://aws.amazon.com/s3/.

[94] J Yates Monteith, John D McGregor, and John E Ingram. “Hadoop and its evolving ecosystem”. In: *5th International Workshop on Software Ecosystems (IWSECO 2013)*. Vol. 50. Citeseer. 2013.

[95] Kathleen Ting and Jarek Jarcec Cecho. “Apache Sqoop Cookbook: Unlocking Hadoop for Your Relational Database”. " O’Reilly Media, Inc.", 2013.

[96] Nishant Garg. “Apache Kafka”. Packt Publishing Ltd, 2013.

[97] A Kala Karun and K Chitharanjan. “A review on hadoop—HDFS infrastructure extensions”. In: *2013 IEEE conference on information & communication technologies*. IEEE. 2013, pp. 132–137.

[98] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. “The Google file system”. In: (2003).

[99] Kristina Chodorow. “MongoDB: the definitive guide: powerful and scalable data storage”. " O’Reilly Media, Inc.", 2013.

[100] Mehul Nalin Vora. “Hadoop-HBase for large-scale data”. In: *Proceedings of 2011 International Conference on Computer Science and Network Technology*. Vol. 1. IEEE. 2011, pp. 601–605.

[101] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[102] Matei Zaharia et al. “Apache spark: a unified engine for big data processing”. In: *Communications of the ACM* 59.11 (2016), pp. 56–65.

[103] Ankit Jain and Anand Nalya. “Learning storm”. Packt Publishing, 2014.

[104] Yichuan Wang and Nick Hajli. “Exploring the path to big data analytics success in healthcare”. In: *Journal of Business Research* 70 (2017), pp. 287–299.

[105] Feras A Batarseh and Eyad Abdel Latif. “Assessing the quality of service using big data analytics: with application to healthcare”. In: *Big Data Research* 4 (2016), pp. 13–24.

[106] Cheng Meng et al. “Effective statistical methods for big data analytics”. In: *Handbook of Research on Applied Cybernetics and Systems Science*. IGI Global, 2017, pp. 280–299.

[107] Victor R Prybutok. “An introduction to statistical methods and data analysis”. 1989.

[108] Changwon Yoo, Luis Ramirez, and Juan Liuzzi. “Big data analysis using modern statistical and machine learning methods in medicine”. In: *International neurourology journal* 18.2 (2014), p. 50.

[109] Tyson Condie et al. “Machine learning on big data”. In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE. 2013, pp. 1242–1244.

[110] Ian H Witten et al. “Data Mining: Practical machine learning tools and techniques”. Morgan Kaufmann, 2016.

[111] “Marketing Cloud Platform Overview.” `https://www.salesforce.com/au/products/marketing-cloud/platform/`.

[112] “BrandsEye.” `https://www.brandseye.com/`.

[113] Paul Rubens. “Big Data + SaaS = Better Marketing.” `https://www.brandseye.com`. March 26, 2016.

[114] Holloway Melanie. “Service level management in cloud computing: Pareto-efficient negotiations, reliable monitoring, and robust monitor placement”. Springer, 2017.

[115] “Gartner Glossary.” `https://www.gartner.com/en/information-technology/glossary/sla-service-level-agreement/`.

[116] “Understanding service level agreements.” `https://www.ibm.com/support/knowledgecenter/en/SS9H2Y_7.6.0/com.ibm.dp.doc/sla_understanding_wsp.html/`.

[117] Tanaji D Dabade. “Information technology infrastructure library (ITIL)”. In: *Proceedings of the 4th National Conference*. 2012, pp. 25–26.

[118] Pankesh Patel, Ajith H Ranabahu, and Amit P Sheth. "Service level agreement in cloud computing". In: (2009).

[119] Christoph Redl et al. "Automatic SLA matching and provider selection in grid and cloud computing markets". In: *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE Computer Society. 2012, pp. 85–94.

[120] Djawida Dib. "Optimizing PaaS provider profit under service level agreement constraints". PhD thesis. 2014.

[121] Kevin Buck and Diane Hanf. "Cloud SLA considerations for the government consumer". In: *Cloud Computing* (2010).

[122] "Cloud service level agreement standardisation guidelines." `http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?action=display&doc_id=6138/`. 2014.

[123] Yaoxue Zhang et al. "A survey on emerging computing paradigms for big data". In: *Chinese Journal of Electronics* 26.1 (2017), pp. 1–12.

[124] Ahmed Patel et al. "Comparative study and review of grid, cloud, utility computing and software as a service for use by libraries". In: *Library Hi Tech News* 28.3 (2011), pp. 25–32.

[125] Paul Wallis. "A brief history of cloud computing: Is the cloud there yet". In: *Cloud Computing Journal* (2008).

[126] Rajleen Kaur and Amanpreet Kaur. "A Review Paper on Evolution of Cloud Computing, its Approaches and Comparison with Grid Computing". In: *International Journal of Computer Science and Information Technologies* 5.5 (2014), pp. 6060–6063.

[127] Rabi Prasad Padhy and Manas Ranjan Patra. "Evolution of cloud computing and enabling technologies". In: *International Journal of Cloud Computing and Services Science* 1.4 (2012), p. 182.

[128] Majid Hajibaba and Saeid Gorgin. "A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing". In: *Journal of computing and information technology* 22.2 (2014), pp. 69–84.

[129] Naidila Sadashiv and SM Dilip Kumar. "Cluster, grid and cloud computing: A detailed comparison". In: *2011 6th International Conference on Computer Science & Education (ICCSE)*. IEEE. 2011, pp. 477–482.

[130] Dr Kahanwal, Dr TP Singh, et al. "The distributed computing paradigms: P2P, grid, cluster, cloud, and jungle". In: *arXiv preprint arXiv:1311.3070* (2013).

[131] Munindar P Singh. "The practical handbook of internet computing". CRC press, 2004.

[132] M Singh. "Peering at peer-to-peer computing". In: *IEEE Internet computing* 5.6 (2001), pp. 4–5.

[133] Rajkumar Buyya et al. "High performance cluster computing: Architectures and systems (volume 1)". In: *Prentice Hall, Upper SaddleRiver, NJ, USA* 1 (1999), p. 999.

[134] Fran Berman, Geoffrey Fox, and Tony Hey. "Overview of the book: grid computing–making the global infrastructure a reality". In: *Making the Global Infrastructure a Reality* (2003), p. 3.

[135] Michael A Rappa. "The utility business model and the future of computing services". In: *IBM systems journal* 43.1 (2004), pp. 32–42.

[136] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. "Big data and cloud computing: current state and future opportunities". In: *Proceedings of the 14th International Conference on Extending Database Technology*. ACM. 2011, pp. 530–533.

[137] Gregory R Andrews. "Foundations of multithreaded, parallel, and distributed programming". Vol. 11. Addison-Wesley Reading, 2000.

[138] "History of the Internet." `https://en.wikipedia.org/wiki/History_of_the_Internet/`.

[139] Lizhe Wang et al. "Cloud computing: a perspective study". In: *New Generation Computing* 28.2 (2010), pp. 137–146.

[140] Maarten van Steen and Andrew S Tanenbaum. "A brief introduction to distributed systems". In: *Computing* 98.10 (2016), pp. 967–1009.

[141] Ralf Lämmel. "Google's MapReduce programming model—Revisited". In: *Science of computer programming* 70.1 (2008), pp. 1–30.

[142] Dinesh C Verma. "Service level agreements on IP networks". In: *Proceedings of the IEEE* 92.9 (2004), pp. 1382–1388.

[143] Edward James Ellesson et al. "Architecture for supporting service level agreements in an IP network". US Patent 6,459,682. Oct. 2002.

[144] R Jo Gibbens et al. "An approach to service level agreements for IP networks with differentiated services". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 358.1773 (2000), pp. 2165–2182.

[145] Nico Brehm, Jorge Marx Gomez, and Claus Rautenstrauch. "An ERP solution based on web services and peer-to-peer networks for small and medium enterprises". In: *International Journal of Information Systems and Change Management* 1.1 (2006), pp. 99–111.

[146] Ralf Steinmetz and Klaus Wehrle. "Peer-to-peer systems and applications". Vol. 3485. Springer, 2005.

[147] Ankur Gupta and Lalit K Awasthi. "Toward a Quality-of-Service Framework for Peer-to-Peer Applications". In: *International Journal of Distributed Systems and Technologies (IJDST)* 1.3 (2010), pp. 1–23.

[148] Kaiqi Xiong and Harry Perros. "SLA-based resource allocation in cluster computing systems". In: *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE. 2008, pp. 1–12.

[149] Kaiqi Xiong and Sang Suh. "Resource provisioning in SLA-based cluster computing". In: *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer. 2010, pp. 1–15.

[150] Chee Shin Yeo and Rajkumar Buyya. "A taxonomy of market-based resource management systems for utility-driven cluster computing". In: *Software: Practice and Experience* 36.13 (2006), pp. 1381–1419.

[151] Chee Shin Yeo and Rajkumar Buyya. "Service level agreement based allocation of cluster resources: Handling penalty to enhance utility". In: *2005 IEEE International Conference on Cluster Computing*. IEEE. 2005, pp. 1–10.

[152] Adrian Ching, Lionel Sacks, and Paul McKee. "SLA management and resource modelling for grid computing". In: *London Communications Symposium (LCS 2003), London, UK*. 2003.

[153] Christopher J Dawson, Roderick E Legg, and Erik Severinghaus. "Management of grid computing resources based on service level requirements". US Patent App. 11/765,487. Dec. 2008.

[154] Avraham Leff, James T Rayfield, and Daniel M Dias. "Service-level agreements and commercial grids". In: *IEEE Internet Computing* 4 (2003), pp. 44–50.

[155] Karen Appleby et al. "Oceano-SLA based management of a computing utility". In: *2001 IEEE/IFIP International Symposium on Integrated Network Management Proceedings. Integrated Network Management VII. Integrated Management Strategies for the New Millennium (Cat. No. 01EX470)*. IEEE. 2001, pp. 855–868.

[156] Andrew DH Farrell et al. "Performance monitoring of service-level agreements for utility computing using the event calculus". In: *Proceedings. First IEEE International Workshop on Electronic Contracting, 2004.* IEEE. 2004, pp. 17–24.

[157] Irfan Ul Haq. "A framework for SLA-centric service-based utility computing". PhD thesis. 2010.

[158] Mohammed Alhamad, Tharam Dillon, and Elizabeth Chang. "Conceptual SLA framework for cloud computing". In: *4th IEEE International Conference on Digital Ecosystems and Technologies*. IEEE. 2010, pp. 606–610.

[159] Eman Aljoumah et al. "SLA in cloud computing architectures: A comprehensive study". In: *Int. J. Grid Distrib. Comput* 8.5 (2015), pp. 7–32.

[160] Vincent C Emeakaroha et al. "Towards autonomic detection of SLA violations in Cloud infrastructures". In: *Future Generation Computer Systems* 28.7 (2012), pp. 1017–1029.

[161] Funmilade Faniyi and Rami Bahsoon. "A systematic review of service level management in the cloud". In: *ACM Computing Surveys (CSUR)* 48.3 (2016), p. 43.

[162] Walayat Hussain et al. "Formulating and managing viable SLAs in cloud computing from a small to medium service provider's viewpoint: A state-of-the-art review". In: *Information Systems* 71 (2017), pp. 240–259.

[163] Akram Mohamadi and Sedigheh Barani. "A review on approaches in service level agreement in cloud computing environment". In: *2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. IEEE. 2015, pp. 1–4.

[164] Yali Zhao et al. "SLA-based resource scheduling for big data analytics as a service in cloud computing environments". In: *2015 44th International Conference on Parallel Processing*. IEEE. 2015, pp. 510–519.

[165] Yali Zhao et al. "SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments". In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 432–441.

[166] Radhya Sahal, Mohamed H Khafagy, and Fatma A Omara. "A survey on SLA management for cloud computing and cloud-hosted big data analytic applications". In: *International Journal of Database Theory and Application* 9.4 (2016), pp. 107–118.

[167] Mihaela-Catalina Nita et al. "A SLA-based method for big-data transfers with multi-criteria optimization constraints for IaaS". In: *2013 11th RoEduNet International Conference*. IEEE. 2013, pp. 1–6.

[168] Jianguo Ding. "Advances in network management". Auerbach Publications, 2016.

[169] "SLA Management Handbook - Volume 1 - Executive Overview." `https://www.itu.int/md/T05-SG04-050920-TD-WP2-0026/en/`. 2005.

[170] "SLA Management Handbook, GB917, Release 3.1." https://www.tmforum.org/resources/best-practice/gb917-sla-management-handbook-release-3-1/. 2005.

[171] "SLA Management Handbook, Volume 2: Concepts and Principles." https://www.tmforum.org/resources/standard/gb917-sla-management-handbook-volume-2-concepts-and-principles-release-2-5-zip/. 2014.

[172] Thomas Schaaf. "Frameworks for business-driven service level management: a criteria-based comparison of ITIL and NGOSS". In: *2007 2nd IEEE/IFIP International Workshop on Business-Driven IT Management*. IEEE. 2007, pp. 65–74.

[173] Shanhe Yi, Cheng Li, and Qun Li. "A survey of fog computing: concepts, applications and issues". In: *Proceedings of the 2015 workshop on mobile big data*. ACM. 2015, pp. 37–42.

[174] Carla Mouradian et al. "A comprehensive survey on fog computing: State-of-the-art and research challenges". In: *IEEE Communications Surveys & Tutorials* 20.1 (2017), pp. 416–464.

[175] Weisong Shi et al. "Edge computing: Vision and challenges". In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646.

[176] Mahadev Satyanarayanan. "The emergence of edge computing". In: *Computer* 50.1 (2017), pp. 30–39.

[177] Blesson Varghese et al. "Challenges and opportunities in edge computing". In: *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE. 2016, pp. 20–26.

[178] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. "Fog computing: A taxonomy, survey and future directions". In: *Internet of everything*. Springer, 2018, pp. 103–130.

[179] Rui Zhang et al. "Getting your big data priorities straight: a demonstration of priority-based qos using social-network-driven stock recommendation". In: *Proceedings of the VLDB endowment* 7.13 (2014), pp. 1665–1668. ISSN: 2150-8097.

[180] Rajiv Ranjan et al. “Cross-layer cloud resource configuration selection in the big data era”. In: *IEEE Cloud Computing* 3 (2015), pp. 16–22. ISSN: 2325-6095.

[181] “Apache mahout”. URL: `https://mahout.apache.org/` (visited on).

[182] Gianmarco De Francisci Morales and Albert Bifet. “SAMOA: scalable advanced massive online analysis.” In: vol. 16. 1. 2015, pp. 149–153.

[183] A Kala Karun and K Chitharanjan. “A review on hadoop—HDFS infrastructure extensions”. In: *2013 IEEE conference on information & communication technologies*. IEEE. 2013, pp. 132–137.

[184] Akram Mohamadi and Sedigheh Barani. “A review on approaches in service level agreement in cloud computing environment”. In: *2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. IEEE. 2015, pp. 1–4.

[185] Funmilade Faniyi and Rami Bahsoon. “A systematic review of service level management in the cloud”. In: vol. 48. 3. 2016, p. 43.

[186] Walayat Hussain et al. “Formulating and managing viable SLAs in cloud computing from a small to medium service provider’s viewpoint: A state-of-the-art review”. In: vol. 71. 2017, pp. 240–259.

[187] Md Whaiduzzaman et al. “A study on strategic provisioning of cloud computing services”. In: vol. 2014. 2014.

[188] Saad Mubeen et al. “Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study”. In: 2017.

[189] Radhya Sahal, Mohamed H Khafagy, and Fatma A Omara. “A Survey on SLA Management for Cloud Computing and Cloud-Hosted Big Data Analytic Applications”. In: vol. 9. 4. 2016, pp. 107–118.

[190] Barbara Kitchenham et al. “Systematic literature reviews in software engineering - a systematic literature review”. In: *Information and software technology* 51.1 (2009), pp. 7–15.

[191] Djawida Dib, Nikos Parlavantzas, and Christine Morin. "SLA-based profit optimization in cloud bursting PaaS". In: *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 141–150. ISBN: 1479927848.

[192] Xiaojun Cai et al. "SLA-aware energy-efficient scheduling scheme for Hadoop YARN". In: vol. 73. 8. 2017, pp. 3526–3546.

[193] "Yarn Scheduler Load Simulator (SLS)". June 2014. URL: `https://hadoop.apache.org/docs/r2.4.1/hadoop-sls/SchedulerLoadSimulator.html/`.

[194] Liang Zhao, Sherif Sakr, and Anna Liu. "A framework for consumer-centric SLA management of cloud-hosted databases". In: vol. 8. 4. 2015, pp. 534–549.

[195] Zhi-guang Ao et al. "Research on Cloud Resource Optimization Model Based on Users' Satisfaction". In: *Web Information Systems and Applications Conference, 2016 13th*. IEEE, 2016, pp. 99–102. ISBN: 1509054375.

[196] Amit Kumar Das et al. "Big media healthcare data processing in cloud: a collaborative resource management perspective". In: vol. 20. 2. 2017, pp. 1599–1614.

[197] Lekha R Nair, Sujala D Shetty, and Siddhanth D Shetty. "Applying spark based machine learning model on streaming big data for health status prediction". In: vol. 65. 2018, pp. 393–399.

[198] Prasan Kumar Sahoo, Suvendu Kumar Mohapatra, and Shih-Lin Wu. "SLA based healthcare big data analysis and computing in cloud network". In: vol. 119. 2018, pp. 121–135.

[199] B Kezia Rani and A Vinaya Babu. "Scheduling of Big Data application workflows in cloud and inter-cloud environments". In: *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 2862–2864. ISBN: 1479999261.

[200] Rajinder Sandhu and Sandeep K Sood. "Scheduling of big data applications on distributed cloud based on QoS parameters". In: vol. 18. 2. 2015, pp. 817–828.

[201] Yanling Shao et al. "Efficient jobs scheduling approach for big data applications". In: vol. 117. 2018, pp. 249–261.

[202] Qinghua Lu et al. “A genetic algorithm-based job scheduling model for big data analytics”. In: vol. 2016. 1. 2016, p. 152.

[203] Qinghua Lu et al. “Autonomic deployment decision making for big data analytics applications in the cloud”. In: vol. 21. 16. 2017, pp. 4501–4512.

[204] Mingruo Shi and Ruiping Yuan. “MAD: A monitor system for big data applications”. In: *International Conference on Intelligent Science and Big Data Engineering*. Springer, 2015, pp. 308–315.

[205] M Omair Shafiq and Eric Torunski. “Towards MapReduce based Bayesian deep learning network for monitoring big data applications”. In: *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2112–2121. ISBN: 1538627159.

[206] Jin Shao and Qianxiang Wang. “A performance guarantee approach for cloud applications based on monitoring”. In: *2011 35th IEEE Annual Computer Software and Applications Conference Workshops*. IEEE, 2011, pp. 25–30. ISBN: 0769544592.

[207] Khalid Alhamazani et al. “Real-time qos monitoring for cloud-based big data analytics applications in mobile environments”. In: *Mobile Data Management (MDM), 2014 IEEE 15th International Conference on*. Vol. 1. IEEE, 2014, pp. 337–340. ISBN: 1479957054.

[208] Rizwan Mian, Patrick Martin, and Jose Luis Vazquez-Poletti. “Provisioning data analytic workloads in a cloud”. In: vol. 29. 6. 2013, pp. 1452–1458.

[209] Mauro Andreolini et al. “Adaptive, scalable and reliable monitoring of big data on clouds”. In: *Journal of Parallel and Distributed Computing* 79 (2015), pp. 67–79.

[210] Maria Krotsiani, Christos Kloukinas, and George Spanoudakis. “Validation of Service Level Agreements using Probabilistic Model Checking”. In: *Services Computing (SCC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 148–155. ISBN: 1538620057.

[211] Yali Zhao et al. "SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments". In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 2016, pp. 432–441.

[212] Damián Serrano et al. "SLA guarantees for cloud services". In: vol. 54. 2016, pp. 233–246.

[213] Yali Zhao et al. "SLA-based resource scheduling for big data analytics as a service in cloud computing environments". In: *2015 44th International Conference on Parallel Processing*. IEEE. 2015, pp. 510–519.

[214] Xuezhi Zeng et al. "Cost efficient scheduling of MapReduce applications on public clouds". In: *Journal of computational science* 26 (2018), pp. 375–388.

[215] Xuezhi Zeng et al. "SLA-aware scheduling of Map-Reduce applications on public clouds". In: *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE. 2016, pp. 655–662.

[216] Deveeshree Nayak et al. "Adaptive scheduling in the cloud-SLA for Hadoop job scheduling". In: *2015 Science and Information Conference (SAI)*. IEEE. 2015, pp. 832–837.

[217] Bikash Sharma, Timothy Wood, and Chita R Das. "Hybridmr: A hierarchical mapreduce scheduler for hybrid data centers". In: *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE. 2013, pp. 102–111.

[218] S Hemalatha and S Valarmathi. "Efficient Hybrid framework for parallel Resource and task scheduling in the Map reduce programming". In: *2016 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE. 2016, pp. 1–7.

[219] Yang Wang and Wei Shi. "On optimal budget-driven scheduling algorithms for MapReduce jobs in the hetereogeneous cloud". In: *Technical Report TR-13–02, Carleton Univ.* (2013).

[220] Qin Zheng. "Improving MapReduce fault tolerance in the cloud". In: *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*. IEEE. 2010, pp. 1–6.

[221] Mihaly Berekmeri et al. "A control approach for performance of big data systems". In: vol. 47. 3. 2014, pp. 152–157.

[222] Xiaoyong Xu, Maolin Tang, and Yu-Chu Tian. "Theoretical results of QoS-guaranteed resource scaling for cloud-based MapReduce". In: 2016.

[223] Xuanjia Qiu et al. "Cost-minimizing preemptive scheduling of mapreduce workloads on hybrid clouds". In: *2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*. IEEE. 2013, pp. 1–6.

[224] Xijun Mao et al. "Optimal Scheduling Algorithm of MapReduce Tasks Based on QoS in the Hybrid Cloud". In: *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2016 17th International Conference on*. IEEE, 2016, pp. 119–124. ISBN: 1509050817.

[225] Damián Serrano et al. "Towards qos-oriented sla guarantees for online cloud services". In: *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*. IEEE. 2013, pp. 50–57.

[226] Palden Lama and Xiaobo Zhou. "Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud". In: *Proceedings of the 9th international conference on Autonomic computing*. ACM. 2012, pp. 63–72.

[227] Ibrahim Abaker Targio Hashem et al. "Multi-objective scheduling of MapReduce jobs in big data processing". In: vol. 77. 8. 2018, pp. 9979–9994.

[228] Yang Wang and Wei Shi. "Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds". In: vol. 2. 3. 2014, pp. 306–319.

[229] Xiaoyong Xu, Maolin Tang, and Yu-Chu Tian. "QoS-guaranteed resource provisioning for cloud-based MapReduce in dynamical environments". In: vol. 78. 2018, pp. 18–30.

[230] Fengguang Tian and Keke Chen. "Towards optimal resource provisioning for running mapreduce programs in public clouds". In: *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 155–162. ISBN: 1457708361.

[231] Balaji Palanisamy, Aameek Singh, and Ling Liu. "Cost-effective resource provisioning for mapreduce in a cloud". In: vol. 26. 5. 2015, pp. 1265–1279.

[232] Michael Mattess, Rodrigo N Calheiros, and Rajkumar Buyya. "Scaling mapreduce applications across hybrid clouds to meet soft deadlines". In: *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2013, pp. 629–636. ISBN: 146735550X.

[233] Abhishek Verma, Ludmila Cherkasova, and Roy H Campbell. "ARIA: automatic resource inference and allocation for mapreduce environments". In: *Proceedings of the 8th ACM international conference on Autonomic computing*. ACM, 2011, pp. 235–244. ISBN: 1450306071.

[234] Lin Gu et al. "Cost minimization for big data processing in geo-distributed data centers". In: vol. 2. 3. IEEE, 2014, pp. 314–323.

[235] S Yasmin and S Jessica Sritha. "A constraint programming-based resource allocation and scheduling of map reduce jobs with service level agreement". In: *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE. 2017, pp. 3589–3594.

[236] Eunji Hwang and Kyong Hoon Kim. "Minimizing cost of virtual machines for deadline-constrained mapreduce applications in the cloud". In: *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE Computer Society, 2012, pp. 130–138. ISBN: 0769548156.

[237] Jorda Polo et al. "Deadline-based MapReduce workload management". In: vol. 10. 2. 2013, pp. 231–244.

[238] KR Krish et al. "Towards energy awareness in hadoop". In: *Network-Aware Data Management (NDM), 2014 Fourth International Workshop on*. IEEE, 2014, pp. 16–22. ISBN: 1479970190.

[239] Chen Wang et al. "Just satisfactory resource provisioning for parallel applications in the cloud". In: *2012 IEEE Eighth World Congress on Services*. IEEE, 2012, pp. 285–292. ISBN: 1467330531.

[240] Adam Gregory and Shikharesh Majumdar. "A configurable energy aware resource management technique for optimization of performance and energy consumption on clouds". In: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016, pp. 184–192. ISBN: 1509014454.

[241] Adam Gregory and Shikharesh Majumdar. "Energy aware resource management for MapReduce jobs with service level agreements in cloud data centers". In: *2016 IEEE International Conference on Computer and Information Technology (CIT)*. IEEE. 2016, pp. 568–577.

[242] Adam Gregory and Shikharesh Majumdar. "A constraint programming based energy aware resource management middleware for clouds processing MapReduce jobs with deadlines". In: *Companion Publication for ACM/SPEC on International Conference on Performance Engineering*. ACM, 2016, pp. 15–20. ISBN: 1450341470.

[243] Norman Lim, Shikharesh Majumdar, and Peter Ashwood-Smith. "A Run Time Technique for Handling Error in User-Estimated Execution Times on Systems Processing MapReduce Jobs with Deadlines". In: *Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on*. IEEE, 2017, pp. 1–9. ISBN: 153862074X.

[244] Norman Lim, Shikharesh Majumdar, and Peter Ashwood-Smith. "Techniques for Handling Error in User-estimated Execution Times During Resource Management on Systems Processing MapReduce Jobs". In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 788–793. ISBN: 1509066101.

[245] Norman Lim, Shikharesh Majumdar, and Peter Ashwood-Smith. "Resource management techniques for handling requests with service level agreements".

In: *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014)*. IEEE. 2014, pp. 618–625.

[246] Norman Lim, Shikharesh Majumdar, and Peter Ashwood-Smith. "MRCP-RM: a technique for resource allocation and scheduling of MapReduce jobs with deadlines". In: vol. 28. 5. 2017, pp. 1375–1389.

[247] Norman Lim, Shikharesh Majumdar, and Peter Ashwood-Smith. "A constraint programming-based resource management technique for processing MapReduce jobs with SLAs on clouds". In: *Parallel Processing (ICPP), 2014 43rd International Conference on*. IEEE, 2014, pp. 411–421. ISBN: 147995618X.

[248] Norman Lim, Shikharesh Majumdar, and Peter Ashwood-Smith. "Engineering resource management middleware for optimizing the performance of clouds processing mapreduce jobs with deadlines". In: *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*. ACM, 2014, pp. 161–172. ISBN: 1450327338.

[249] Sofia D'Souza and K Chandrasekaran. "Analysis of MapReduce scheduling and its improvements in cloud environment". In: *Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–5. ISBN: 1479918237.

[250] Eugenio Gianniti et al. "A game-theoretic approach for runtime capacity allocation in MapReduce". In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 1080–1089. ISBN: 1509066101.

[251] Banpreet Kaur and Ankit Grover. "Optimizing VM Provisioning of MapReduce Tasks on Public Cloud". In: *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*. ACM, 2016, p. 79. ISBN: 1450342132.

[252] Yoori Oh et al. "A SLA-based Spark cluster scaling method in cloud environment". In: *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE. 2016, pp. 1–4.

[253] Anshul Gandhi et al. “Autoscaling for Hadoop clusters”. In: *2016 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2016, pp. 109–118. ISBN: 1509019618.

[254] Yang Lu. “Industry 4.0: A survey on technologies, applications and open research issues”. In: vol. 6. 2017, pp. 1–10.

[255] Mihaela-Catalina Nita et al. “MOMTH: multi-objective scheduling algorithm of many tasks in Hadoop”. In: vol. 18. 3. 2015, pp. 1011–1024.

[256] Yanling Shao et al. “Energy-aware dynamic resource allocation on Hadoop YARN cluster”. In: *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*. IEEE, 2016, pp. 364–371. ISBN: 1509042970.

[257] Lena Mashayekhy et al. “Energy-aware scheduling of mapreduce jobs for big data applications”. In: 1. 2015, pp. 1–1.

[258] Selvi Kadirvel and José AB Fortes. “Towards self-caring mapreduce: Proactively reducing fault-induced execution-time penalties”. In: *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE, 2011, pp. 63–71. ISBN: 1612843832.

[259] Mohammed Alrokayan, Amir Vahid Dastjerdi, and Rajkumar Buyya. “Sla-aware provisioning and scheduling of cloud resources for big data analytics”. In: *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE. 2014, pp. 1–8.

[260] Paolo Bellavista et al. “Priority-based resource scheduling in distributed stream processing systems for big data applications”. In: *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2014, pp. 363–370. ISBN: 1479978817.

[261] Radu Tudoran et al. “Jetstream: Enabling high performance event streaming across cloud data-centers”. In: *Proceedings of the 8th ACM International*

Conference on Distributed Event-Based Systems. ACM, 2014, pp. 23–34. ISBN: 1450327370.

[262] Panya Kittipipattanathaworn and Natawut Nupairoj. "SLA guarantee real-time monitoring system with soft deadline constraint". In: *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on*. IEEE, 2014, pp. 52–57. ISBN: 1479958220.

[263] Shigeru Imai, Stacy Patterson, and Carlos A Varela. "Maximum sustainable throughput prediction for data stream processing over public clouds". In: *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE Press, 2017, pp. 504–513. ISBN: 1509066101.

[264] Linjiun Tsai et al. "Learning-Based Memory Allocation Optimization for Delay-Sensitive Big Data Processing". In: vol. 29. 6. 2018, pp. 1332–1341.

[265] Rafael Tolosana-Calasanz et al. "Resource management for bursty streams on multi-tenancy cloud environments". In: vol. 55. Elsevier, 2016, pp. 444–459.

[266] Christoph Hochreiner et al. "Cost-efficient enactment of stream processing topologies". In: vol. 3. 2017, e141.

[267] Lin Gu et al. "A general communication cost optimization framework for big data stream processing in geo-distributed data centers". In: vol. 65. 1. 2016, pp. 19–29.

[268] Andre Abrantes DP Souza and Marco AS Netto. "Using application data for sla-aware auto-scaling in cloud environments". In: *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on*. IEEE, 2015, pp. 252–255. ISBN: 1467377201.

[269] Muhammad Hanif et al. "An adaptive SLA-based data flow mechanism for stream processing engines". In: *Information and Communication Technology Convergence (ICTC), 2017 International Conference on*. IEEE, 2017, pp. 81–86. ISBN: 1509040323.

[270] Ming-Hung Shih and J Morris Chang. "Design and analysis of high performance crypt-NoSQL". In: *Dependable and Secure Computing, 2017 IEEE Conference on*. IEEE, 2017, pp. 52–59. ISBN: 150905569X.

[271] Victor AE Farias et al. "Regression based performance modeling and provisioning for NoSQL cloud databases". In: vol. 79. 2018, pp. 72–81.

[272] Dimas C Nascimento, Carlos Eduardo Pires, and Demetrio Mestre. "Data quality monitoring of cloud databases based on data quality slas". In: *Big-Data Analytics and Cloud Computing*. Springer, 2015, pp. 3–20.

[273] Chao-Wen Huang et al. "The improvement of auto-scaling mechanism for distributed database-A case study for MongoDB". In: *2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE. 2013, pp. 1–3.

[274] Sherif Sakr and Anna Liu. "Sla-based and consumer-centric dynamic provisioning for cloud databases". In: *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 360–367. ISBN: 1467328928.

[275] Tadeusz Pankowski. "Consistency and availability of Data in replicated NoSQL databases". In: *Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 International Conference on*. IEEE, 2015, pp. 102–109. ISBN: 989758143X.

[276] Ramon Hugo de Souza et al. "Architectural recovering model for Distributed Databases: A reliability, availability and serviceability approach". In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE. 2016, pp. 575–580.

[277] Carla Sauvanaud et al. "Anomaly Detection and Diagnosis for Cloud services: Practical experiments and lessons learned". In: *Journal of Systems and Software* 139 (2018), pp. 84–106.

[278] Sanping Li et al. "An extensible framework for predictive analytics on cost and performance in the cloud". In: *2015 International Conference on Cloud Computing and Big Data (CCBD)*. IEEE, 2015, pp. 13–20. ISBN: 1467383503.

[279] Mohammad-Hossein Malekloo, Nadjia Kara, and May El Barachi. "An energy efficient and SLA compliant approach for resource allocation and consolidation in cloud computing environments". In: vol. 17. 2018, pp. 9–24.

[280] Omran Saleh et al. "Monitoring and autoscaling IaaS clouds: a case for complex event processing on data streams". In: *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE Computer Society. 2013, pp. 387–392.

[281] Fotios K Liotopoulos and Petros Lampsas. "Energy-efficient simulation and performance evaluation of large-scale data centers". In: *2015 IEEE International Conference on Industrial Technology (ICIT)*. IEEE. 2015, pp. 3121–3127.

[282] Hyejeong Kang et al. "A SLA driven VM auto-scaling method in hybrid cloud environment". In: *Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific*. IEEE, 2013, pp. 1–6. ISBN: 488552279X.

[283] Yeongho Choi and Yujin Lim. "Resource management mechanism for SLA provisioning on cloud computing for IoT". In: *2015 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2015, pp. 500–502.

[284] Yeongho Choi and Yujin Lim. "Optimization approach for resource allocation on cloud computing for iot". In: vol. 2016. 2016, p. 23.

[285] Ali Imran Jehangiri et al. "Diagnosing cloud performance anomalies using large time series dataset analysis". In: *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 930–933. ISBN: 1479950637.

[286] I-Hsun Chuang et al. "S4: An SLA-aware Short-Secret-Sharing cloud storage system". In: *2015 Seventh International Conference on Ubiquitous and Future Networks*. IEEE. 2015, pp. 401–406.

[287] Joy Rahman and Palden Lama. "MPLEX: In-Situ Big Data Processing with Compute-Storage Multiplexing". In: *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2017 IEEE 25th International Symposium on*. IEEE, 2017, pp. 43–52. ISBN: 1538627647.

[288] Zhihao Yao, Ioannis Papapanagiotou, and Robert D Callaway. "SLA-aware resource scheduling for cloud storage". In: *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 14–19. ISBN: 1479927309.

[289] Mouhamad Dieye, Mohamed Faten Zhani, and Halima Elbiaze. "On achieving high data availability in heterogeneous cloud storage systems". In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 326–334.

[290] Ahmad B Alnafoosi and Theresa Steinbach. "An integrated framework for evaluating big-data storage solutions-IDA case study". In: *Science and Information Conference (SAI), 2013*. IEEE, 2013, pp. 947–956. ISBN: 098931930X.

[291] Yangyang Tao, Shucheng Yu, and Junxiu Zhou. "Information Flow Queue Optimization in EC Cloud". In: *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2018, pp. 888–892. ISBN: 1538636522.

[292] Mihaela-Catalina Nita et al. "A SLA-based method for big-data transfers with multi-criteria optimization constraints for IaaS". In: *2013 11th RoEduNet International Conference*. IEEE. 2013, pp. 1–6.

[293] Liudong Zuo and Michelle M Zhu. "Concurrent bandwidth reservation strategies for big data transfers in high-performance networks". In: vol. 12. 2. 2015, pp. 232–247.

[294] Abdulsalam Yassine, Ali Asghar Nazari Shirehjini, and Shervin Shirmohammadi. "Bandwidth on-demand for multimedia big data transfer across geo-distributed cloud data centers". In: 2016.

[295] Xiaoqun Yuan et al. "A game theory-based dynamic resource allocation strategy in geo-distributed datacenter clouds". In: vol. 76. 2017, pp. 63–72.

[296] Pham Phuoc Hung et al. "A new technique for optimizing resource allocation and data distribution in mobile cloud computing". In: vol. 22. 1. 2016, pp. 73–80.

[297] K Hima Prasad et al. “Resource allocation and SLA determination for large data processing services over cloud”. In: *2010 IEEE International Conference on Services Computing*. IEEE, 2010, pp. 522–529. ISBN: 0769541267.

[298] Karim Kanoun et al. “Big-data streaming applications scheduling based on staged multi-armed bandits”. In: vol. 65. 12. 2016, pp. 3591–3605.

[299] William H Bell et al. “Simulation of dynamic grid replication strategies in optorsim”. In: *International Workshop on Grid Computing*. Springer. 2002, pp. 46–57.

[300] Xi Sun et al. “A cost-effective approach to delivering analytics as a service”. In: *Web services (icws), 2012 ieee 19th international conference on*. IEEE, 2012, pp. 512–519. ISBN: 1467321311.

[301] Laouratou Diallo et al. “Two objectives big data task scheduling using swarm intelligence in cloud computing”. In: vol. 9. 28. 2016.

[302] Rajkumar Buyya et al. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”. In: vol. 25. 6. 2009, pp. 599–616.

[303] Mbarka Soualhia, Foutse Khomh, and Sofiène Tahar. “Task scheduling in big data platforms: a systematic literature review”. In: vol. 134. Elsevier, 2017, pp. 170–189.

[304] Bernard P Zeigler, Tag Gon Kim, and Herbert Praehofer. “Theory of modeling and simulation”. Academic press, 2000.

[305] Raymond Hoare, Jiyong Ahn, and Jesse Graves. “Discrete event simulator”. In: Google Patents, 2002 of Conference.

[306] Guanying Wang et al. “Using realistic simulation for performance analysis of mapreduce setups”. In: *Proceedings of the 1st ACM workshop on Large-Scale system and application performance*. ACM, 2009, pp. 19–26. ISBN: 1605585920.

[307] Jonathan Stuart Ward and Adam Barker. “Observing the clouds: a survey and taxonomy of cloud monitoring”. In: vol. 3. 1. 2014, p. 24.

[308] Adrian Paschke and Elisabeth Schnappinger-Gerull. “A Categorization Scheme for SLA Metrics”. In: vol. 80. 25-40. 2006, p. 14.

[309] Mohammed Alhamad, Tharam Dillon, and Elizabeth Chang. “Conceptual SLA framework for cloud computing”. In: *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*. IEEE, 2010, pp. 606–610. ISBN: 1424455537.

[310] Taher Labidi, Achraf Mtibaa, and Hayet Brabra. “CSLAOnto: a comprehensive ontological SLA model in cloud computing”. In: vol. 5. 3. 2016, pp. 179–193.

[311] Jonathan Lejeune, Frederico Alvares, and Thomas Ledoux. “Towards a generic autonomic model to manage Cloud Services”. In: *The 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*. 2017.

[312] Lucia De Marco, Filomena Ferrucci, and Tahar Kechadi. “SLAFM: A Service Level Agreements Formal Model for Cloud Computing”. In: *The 5th International Conference on Cloud Computing and Service Science (CLOSER 2015), Lisbon, Portugal, 20-22 May 2015*. 2015.

[313] Todd Escalona. “Detect sentiment from customer reviews using Amazon Comprehend”. January 2018. URL: `https://aws.amazon.com/blogs/machine-learning/detect-sentiment-from-customer-reviews-using-amazon-comprehend/` (visited on).

[314] Gabriel Iuhasz and Ioan Dragan. “An overview of monitoring tools for big data and cloud applications”. In: *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2015 17th International Symposium on*. IEEE, 2015, pp. 363–366. ISBN: 1509004610.

[315] Meisong Wang et al. “A Case for Understanding End-to-End Performance of Topic Detection and Tracking Based Big Data Applications in the Cloud”. In: *International Internet of Things Summit*. Springer. 2015, pp. 315–325.

[316] Tao Cheng and Thomas Wicks. “Event detection using Twitter: a spatio-temporal approach”. In: vol. 9. 6. 2014, e97807.

[317] Néstor Cárdenas-Benıtez et al. "Traffic congestion detection system through connected vehicles and big data". In: *Sensors* 16.5 (2016), p. 599.

[318] "Architecture: Optimizing Large-Scale Ingestion of Analytics Events and Logs". October 2018. URL: https://cloud.google.com/solutions/architecture/optimized-large-scale-analytics-ingestion/ (visited on).

[319] "Google Compute Engine Service Level Agreement (SLA)". March 2019. URL: https://cloud.google.com/compute/sla/ (visited on).

[320] "Google Cloud Storage Service Level Agreement (SLA)". October 2018. URL: https://cloud.google.com/storage/sla/ (visited on).

[321] "Google VPN Service Level Agreement (SLA)". October 2016. URL: https://cloud.google.com/vpn/sla/ (visited on).

[322] "Google App Engine Service Level Agreement (SLA)". October 2016. URL: https://cloud.google.com/appengine/sla/ (visited on).

[323] "Google Cloud SQL Service Level Agreement (SLA)". April 2017. URL: https://cloud.google.com/sql/sla/ (visited on).

[324] Allan Gibb et al. "A comparison of deterministic vs stochastic simulation models for assessing adaptive information management techniques over disadvantaged tactical communication networks". In: *7th ICCRTS* (2002), pp. 1–16.

[325] Theodore T Allen. "Introduction to discrete event simulation and agent-based modeling: voting systems, health care, military, and manufacturing". Springer Science & Business Media, 2011.

[326] Donald C Craig. "Extensible hierarchical object-oriented logic simulation with an adaptable graphical user interface". PhD thesis. Memorial University of Newfoundland, 1996.

[327] Anu Maria. "Introduction to modeling and simulation". In: *Winter simulation conference*. Vol. 29. 1. 1997, pp. 7–13.

[328] Dirk P Kroese and Reuven Y Rubinstein. "Monte carlo methods". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.1 (2012), pp. 48–58.

[329] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers". In: *The Journal of Supercomputing* 62.3 (2012), pp. 1263–1283.

[330] Alberto Núñez et al. "iCanCloud: A flexible and scalable cloud infrastructure simulator". In: *Journal of Grid Computing* 10.1 (2012), pp. 185–209.

[331] Minxian Xu et al. "Flexcloud: A flexible and extendible simulator for performance evaluation of virtual machine allocation". In: *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*. IEEE. 2015, pp. 649–655.

[332] Rodrigo N Calheiros et al. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". In: *Software: Practice and experience* 41.1 (2011), pp. 23–50.

[333] Bhathiya Wickremasinghe, Rodrigo N Calheiros, and Rajkumar Buyya. "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications". In: *2010 24th IEEE international conference on advanced information networking and applications*. IEEE. 2010, pp. 446–452.

[334] Saurabh Kumar Garg and Rajkumar Buyya. "Networkcloudsim: Modelling parallel applications in cloud simulations". In: *2011 Fourth IEEE International Conference on Utility and Cloud Computing*. IEEE. 2011, pp. 105–113.

[335] Rodrigo N Calheiros et al. "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications". In: *Software: Practice and Experience* 43.5 (2013), pp. 595–612.

[336] Seung-Hwan Lim et al. "MDCSim: A multi-tier data center simulation, platform." In: *CLUSTER*. Vol. 31. 2009, pp. 1–9.

[337] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities". In: *2009 international conference on high performance computing & simulation*. IEEE. 2009, pp. 1–11.

[338] Suhel Hammoud et al. "MRSim: A discrete event based MapReduce simulator". In: *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 6. IEEE. 2010, pp. 2993–2997.

[339] Arun C Murthy. "Mumak: Map-reduce simulator". In: *MAPREDUCE-728, Apache JIRA* (2009).

[340] Abhishek Verma, Ludmila Cherkasova, and Roy H Campbell. "Play it again, simmr!" In: *2011 IEEE International Conference on Cluster Computing*. IEEE. 2011, pp. 253–261.

[341] Costas Simatos. "Making simjava count". In: *MSc. Project report, The University of Edinburgh* (2002).

[342] Jongtack Jung and Hwangnam Kim. "MR-CloudSim: Designing and implementing MapReduce computing model on CloudSim". In: *2012 International Conference on ICT Convergence (ICTC)*. IEEE. 2012, pp. 504–509.

[343] Abul Bashar. "Modeling and simulation frameworks for cloud computing environment: A critical evaluation". In: *International Conference on Cloud Computing and Services Science*. 2014, pp. 1–6.

[344] Gunho Lee and Randy H Katz. "Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud." In: *HotCloud*. 2011.

[345] Karthik Kambatla, Abhinav Pathak, and Himabindu Pucha. "Towards Optimizing Hadoop Provisioning in the Cloud." In: *HotCloud* 9.12 (2009), pp. 28–30.

[346] Herodotos Herodotou and Shivnath Babu. "A What-if Engine for Cost-based MapReduce Optimization." In: *IEEE Data Eng. Bull.* 36.1 (2013), pp. 5–14.

[347] Silvano Martello. "An algorithm for the generalized assignment problem". In: *Operational research* (1981), pp. 589–603.

[348] Abhishek Verma, Ludmila Cherkasova, and Roy H Campbell. "Resource provisioning framework for mapreduce jobs with performance goals". In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2011, pp. 165–186.

[349] Philipp Leitner et al. "Runtime prediction of service level agreement violations for composite services". In: *Service-oriented computing. ICSOC/ServiceWave 2009 workshops*. Springer, 2010, pp. 176–186.

[350] Obed Jules, Abdelhakim Hafid, and Mohamed Adel Serhani. "Bayesian network, and probabilistic ontology driven trust model for sla management of cloud services". In: *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 77–83. ISBN: 1479927309.

[351] Bing Tang and Mingdong Tang. "Bayesian model-based prediction of service level agreement violations for cloud services". In: *Theoretical Aspects of Software Engineering Conference (TASE), 2014*. IEEE, 2014, pp. 170–176. ISBN: 1479950297.

[352] Reyhane Askari Hemmat and Abdelhakim Hafid. "SLA violation prediction in cloud computing: A machine learning perspective". In: *arXiv preprint arXiv:1611.10338* (2016).

[353] "Google Cluster Workload Traces". 2013. URL: `https://github.com/google/cluster-data` (visited on).

[354] Rafael Brundo Uriarte, Sotirios Tsaftaris, and Francesco Tiezzi. "Service clustering for autonomic clouds using random forest". In: *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE, 2015, pp. 515–524. ISBN: 1479980064.

[355] "Alibaba Cluster Trace v2017". September 2017. URL: `https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2017` (visited on).

[356] Chengzhi Lu et al. "Imbalance in the cloud: an analysis on Alibaba cluster trace". In: *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2884–2892. ISBN: 1538627159.

[357] Qixiao Liu and Zhibin Yu. "The Elasticity and Plasticity in Semi-Containerized Co-locating Cloud Workload: a View from Alibaba Trace". In: *Proceedings of ACM Symposium on Cloud Computing (SOCC)*. 2018.

[358] Yue Cheng, Zheng Chai, and Ali Anwar. "Characterizing co-located datacenter workloads: An Alibaba case study". In: *arXiv preprint arXiv:1808.02919* (2018).

[359] Oliver Kramer. "Scikit-learn". In: *Machine learning for evolution strategies*. Springer, 2016, pp. 45–53.

[360] Sandro Sperandei. "Understanding logistic regression analysis". In: *Biochemia medica: Biochemia medica* 24.1 (2014), pp. 12–18.

[361] Anil K Jain, Jianchang Mao, and KM Mohiuddin. "Artificial neural networks: A tutorial". In: *Computer* 3 (1996), pp. 31–44.

[362] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[363] L Torlay et al. "Machine learning–XGBoost analysis of language networks to classify patients with epilepsy". In: *Brain informatics* 4.3 (2017), p. 159.

[364] George Bebis and Michael Georgiopoulos. "Feed-forward neural networks". In: *IEEE Potentials* 13.4 (1994), pp. 27–31.

[365] Tianqi Chen et al. "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* (2015), pp. 1–4.

[366] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. "Handling imbalanced datasets: A review". In: *GESTS International Transactions on Computer Science and Engineering* 30.1 (2006), pp. 25–36.

[367] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[368] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 1322–1328. ISBN: 1424418208.

[369] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning". In: *International Conference on Intelligent Computing*. Springer, 2005, pp. 878–887.

[370] Miroslav Kubat, Stan Matwin, et al. "Addressing the curse of imbalanced training sets: one-sided selection". In: *Icml*. Vol. 97. Nashville, USA. 1997, pp. 179–186.

[371] Inderjeet Mani and I Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction". In: *Proceedings of workshop on learning from imbalanced datasets*. Vol. 126. 2003.

[372] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* 6.1 (2004), pp. 20–29.

[373] Nitesh V Chawla et al. "Automatically countering imbalance and its empirical relationship to cost". In: *Data Mining and Knowledge Discovery* 17.2 (2008), pp. 225–252.

Publications during Ph.D. Study

Some of the work presented in this thesis haven been presented in the following papers.

[1] Xuezhi Zeng et al. "Cross-layer SLA management for cloud-hosted big data analytics applications". In: *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE. 2015, pp. 765–768.

[2] Xuezhi Zeng et al. "IoTSim: A simulator for analysing IoT applications". In: *Journal of Systems Architecture* 72 (2017), pp. 93–107.

[3] Xuezhi Zeng et al. "SLA-aware scheduling of Map-Reduce applications on public clouds". In: *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE. 2016, pp. 655–662.

[4] Xuezhi Zeng et al. "Cost efficient scheduling of MapReduce applications on public clouds". In: *Journal of computational science* 26 (2018), pp. 375–388.

[5] Xuezhi Zeng et al. "SLA Management for Big Data Analytical Applications in Clouds: A Taxonomy Study". In: *ACM Computing Surveys, 2019* (Accepted on October 2019).

[6] Xuezhi Zeng et al. "Detection of SLA Violation for Big Data Analytics Applications in Cloud". In: *Journal of IEEE Transactions on computer,2019* (Under Review).