



HOKKAIDO UNIVERSITY

Title	Multi-Objective Resource Optimization of Big Data Applications in Cloud Computing Infrastructure
Author(s)	Phyo Thandar Thant
Degree Grantor	北海道大学
Degree Name	博士(情報科学)
Dissertation Number	甲第13079号
Issue Date	2018-03-22
DOI	https://doi.org/10.14943/doctoral.k13079
Doc URL	https://hdl.handle.net/2115/70250
Type	doctoral thesis
File Information	Phyo_Thandar_Thant.pdf



Multi-Objective Resource Optimization of Big Data Applications in Cloud Computing Infrastructure

クラウドコンピューティング基盤におけるビッグデータ
アプリケーションの多目的資源割当最適化

Phyo Thandar Thant

A Dissertation

submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Information Science and Technology

in the field of

Computer Science and Information Technology

at the

Graduate School of Information Science and Technology

Hokkaido University, Japan



2018

Abstract

Experimental applications in the scientific area have become very complex with ever increasing amounts of Big Data that highlight the criticality of high- performance computing environments such as clusters, grids and the cloud. Grid computing systems were widely used in scientific applications in the past as they provide parallelism through high-performance computational infrastructure. However, the adoption of cloud computing to process these scientific data has increased recently because it enables researchers to execute workflows at low costs without setting up their own infrastructure. Moreover, the elastic nature of cloud facilitates the changing of resource quantities and characteristics to vary at runtime, facilitating dynamic scaling of infrastructure resources according to application requirements and budget. Further, nowadays, large-scale genome data presents a significant challenge for data processing and analytics in Big Data environment. In this scenario, Galaxy, a widely used scientific workflow management system, provides an open, web-based platform for data-intensive computational analyses and data integration. However, the resources needed by scientific workflows vary significantly during execution on Galaxy and may become inefficient in terms of resource usage and cost. Therefore, researchers are focused on deploying Galaxy in the cloud for better resource utilization and data analytics. Nevertheless, several research issues associated with the deploying of scientific workflow applications in the cloud still exists. Among them, resource selection to make the right choice of instances for a certain application of interest is a challenging problem for researchers. In addition, providing services with optimal performance at the lowest financial resource deployment cost based on users' resource selection is quite challenging for cloud service providers.

Scientific workflows are a series of computation-dependent tasks occurring in scientific experiments. They have emerged as an alternative to script programming for performing in-silico experiments and are concerned with the processing of high-volume scientific

Big Data. Several workflow optimization studies have focused on optimizing makespan through scheduling mechanisms, whereas others have conducted multi-objective optimization during workflow execution. However, conventional workflow optimization approaches only consider makespan and cost optimization and do not cover crucial factors related with virtual machine (VM) instances such as VM failure probability and availability probability, which can significantly affect the service level agreements (SLAs) of the cloud during workflow execution.

This thesis begins with some preliminary studies of Big Data applications in Hadoop infrastructure by using VM instances from cloud computing infrastructure. Subsequently, it focuses on investigation of the challenges of resource optimization of scientific workflow executions in the cloud and develops a new constrained multi-objective optimization model for general scientific workflow applications such as bioinformatics, environmental issues, and astronomy using multi-objective evolutionary algorithms that benefit both stakeholders (users and cloud service providers). The following contributions are made during resource optimization of the proposed system: (i) modeling of the resource optimization problem as the constrained multi-objective optimization problem, (ii) randomized search-based resource optimization for rightsized instance resource selection, (iii) level-wise workflow execution optimization for task dependencies and resource elasticity, (iv) level-wise balanced task clustering and assignment for better load balancing and resource sharing.

Thus, the system provides several advantages such as elimination of the resource selection problem confronting novice cloud users, optimal resource assignment, optimal budget management and improved service satisfaction for cloud users for the service providers. This system can support the rapid adoption of scientific applications to more efficient cloud applications and can also handle output validity problem of large, critical scientific applications in certain situations.

学位論文審査の要旨

科学技術分野におけるビッグデータ処理が近年その重要性を増しており、クラウドコンピューティングの急速な発展にともない様々なクラウドサービスや計算資源を組み合わせ、ゲノム解析ワークフローなどのビッグデータ処理を実現することが可能となりつつある。しかしながら、多岐にわたるサービスや資源を組み合わせ、高性能なシステムを低コストで、かつ信頼性などの制約条件を満たしつつ構築することは困難な課題となっており、その最適化に向けた研究開発がなされてきた。

本論文では、科学技術分野におけるビッグデータ処理において最も一般的に用いられるものとして、**Hadoop** に代表されるビッグデータ処理フレームワークと、ゲノム解析ワークフローなどのサイエンティフィックワークフローを例にとりそれらの処理をプライベートクラウドやパブリッククラウドで提供される仮想マシンを用いて効率的に実現するための、多目的資源割当最適化について議論している。ビッグデータ処理には多くの計算資源および計算時間を要することから、研究者の観点からは、できるだけ短い時間で処理を終えること、その処理に必要なクラウド利用料金のコストを最小化すること、さらには実行時の信頼性についても確保する必要があるなど、複数の評価尺度を同時に考慮した多目的最適化が求められる。また、特にパブリッククラウドでは数多くのインスタンスタイプが提供されるなど、サービスの選択肢が多岐にわたり、また、**Hadoop** などのフレームワークやワークフローの構築方法などに関するパラメータも数多く存在することから、最適なパラメータを手動で選択するのは事実上不可能である。そこで、本論文においては、多目的進化計算アルゴリズムを用いた統一的な枠組みのもとでクラウドコンピューティング基盤におけるビッグデータ処理に関わる多目的資源割当最適化問題を解決する。

第1章では、ビッグデータ処理における多目的最適化の必要性、および本論文において提案する解決策の概要について議論している。

第2章では、関連技術および既存研究として、クラウドコンピューティング、**Hadoop**などビッグデータ処理フレームワーク、サイエンティフィックワークフロー、多目的進化計算アルゴリズムについて議論している。

第3章では、ビッグデータ処理フレームワークとして代表的な **Hadoop** を例にとり、その資源割当およびパラメータ最適化を、実行時間および実行コストの観点から多目的進化計算アルゴリズムを適用した手法について議論している。多目的進化計算アルゴリズムの適用において、その適応度評価に **Hadoop** を実環境で実行する必要がある多くの計算時間を必要とすることから、適応度評価を削減するために集団内の一部の個体について評価を行い置き換えつつ探索をすすめる改良を行った。その結果として、実行時間と実行コストのトレードオフ関係を反映したパレート最適解集合を求めることができ、**Hadoop** の利用者が多岐にわたる複雑なパラメータの設定を手動で行う必要なく、効率的なビッグデータ処理を実現することができた。

第4章では、ゲノム解析処理フローなどのサイエンティフィックワークフローによるビッグデータ処理に関するクラウド資源割当最適化問題に対して、多目的進化計算アルゴリズムを適用することで、実行時間、実行コスト、さらには **SLA (Service Level Agreement)** など信頼性に関する複数の評価尺度を考慮したパレート最適解集合を求めることができた。

第5章、第6章においては、以上の結果に関して考察するとともに、今後に向けた展望について議論している。結果として、提案する多目的進化計算アルゴリズムによる多目的資源割当最適化により、科学技術分野で代表的なビッグデータ処理アプリケーションである **Hadoop**、およびサイエンティフィックワークフローにおいて、複雑かつ多岐にわたるクラウド資源割当最適化およびそのパラメータ設定最適化に関わる問題を、統一的な枠組みにより解決し、得られたパレート最適解集合をもとに研究者がビッグデータ処理を短時間、低コストで、かつ高い信頼性のもとで行える基盤技術を確立したものである。

これを要するに、本論文は先端的な多目的進化計算アルゴリズムを用いるこ

とで、ビッグデータ処理フレームワークにおけるクラウド資源割当・パラメータ最適化、および、サイエンティフィックワークフローに対応するクラウド資源割当の最適化を実現しており、近年その重要性が高まっているビッグデータ処理アプリケーションの効率的な実現に貢献するところ大なるものがある。

Acknowledgments

First, I would like to express my gratitude to the Japanese Government for giving me the opportunity to attend the doctoral degree course at Hokkaido University as a MEXT scholarship student and providing financial assistance to conduct this research.

I would also like to express special thanks to Dr. Masaharu Munetomo, Professor and Vice Director of Information Initiative Center, Hokkaido University. He is my advisor and has provided not only a helpful guidance but also much inspiration, motivation, and encouragement during my research. He was a strong influence in instilling inspiration and motivational drive to tide me over the arduous tasks of research activities through all the stages. I have learnt many things about research working together with him, and his support has been essential for the success of my work.

I would also grateful to Dr. Akiyoshi Sugiki, Associate Professor, Information Systems Design Lab, Information Initiative Center, Hokkaido University for his overall support during my research.

My sincerest gratitude also goes out to the members of my dissertation committee, Professor Yoshiaki Takai and Professor Masahiro Mizuta for providing valuable suggestions and comments that improved my research and dissertation.

I am also thankful to Lecturer Dr. Katsunori Miura (Kitami Institute of Technology), Assistant Professor Dr. Courtney Powell and Dr. Martin Schlueter who helped me in my research. My thanks also goes out to Mr. Bando Shintaro, Mr. Masataka Mizukoshi, Mr. Akahito Seyama, Mr. Tomoya Abe, Mr. Jun Iwatsubo, Mr. Ryohei Ichii, Mr. Atsushi Saito, Mr. Taisuke Okabe, Mr. Tohru Hata, Mr. Kousuke Izumiya, Mr. Ryosei Iwai, Mr. Shunichi Fujita, Mr. Masaki Fujiwara, Mr. Hiroki Kotoda and all my laboratory members who helped me not only research related things but on things related to my daily life in Japan.

I also thank all the reviewers of my published papers in journals, workshops and inter-

national conferences for their useful comments and suggestions that were invaluable in my research.

This work is partially supported by the CREST project and I would like to thank all professors and colleagues in the CREST project as well. Sincere gratitude also goes to my Japanese Language Intensive and General courses' teachers (Prof. Yoshitaka Yamashita, Mariko Nishida Sensei, Noriko Suzuki Sensei, Keiko Mizutani Sensei, Yumi Oonishi Sensei, Matsuko Sudo Sensei, Takayuki Ito Sensei, Satoko Imaizumi Sensei, Junko Homa Sensei and Yoshiro Ogawara Sensei) for their kind support during my Japanese studies. My thanks also go to my teachers from Nitobe School, Hokkaido University to providing self-improvement opportunities that enhanced my mindset during my research.

Finally, I am grateful to my father Mr. Maung Maung Thaung, my mother Mrs. Khin Aye Kyi, my siblings (Dr. Shwe Zin Mon, Dr. Zaw Linn Htat, Ms. Khin Thet Htar Wai), my teachers (especially Dr. Mie Mie Thet Thwin, Dr. Saw Sanda Aye, Dr. Thinn Thu Naing), and my friends from Myanmar for their strong moral and physical support, care, and kindness, throughout my Ph.D. journey. Without their full support, my dissertations would not have been possible.

Contents

Abstract	i
Acknowledgements	vi
Table of Contents	viii
1 Introduction	1
1.1 Motivation for this Thesis	2
1.2 Objectives of this Thesis	3
1.3 Contributions of this Thesis	4
1.4 Problem Origin of the System	5
1.5 Overview of Multi-Objective Resource Optimization of Big Data Applica- tions	5
1.6 Benefits of the System	7
1.7 Organization of this Thesis	7
2 Background	9
2.1 Cloud Computing	9
2.2 Features of Cloud Computing	10
2.3 Cloud Deployment Models	11
2.3.1 Public Cloud	11
2.3.2 Private Cloud	12

2.3.3	Community Cloud	12
2.3.4	Hybrid Cloud	13
2.4	Cloud Service Layers	13
2.4.1	Infrastructure as a Service (IaaS)	13
2.4.2	Platform as a Service (PaaS)	14
2.4.3	Software as a Service (SaaS)	14
2.5	Big Data	15
2.6	Challenges of Big Data	16
2.7	Cloud Computing and Big Data	17
2.8	Hadoop and Big Data	18
2.9	Scientific Workflows	18
2.10	Cloud Computing and Scientific Workflows	20
2.11	Multi-Objective Evolutionary Optimization (MOEA)	21
2.11.1	The Simple Evolutionary Multi-Objective Optimizer (SEMO)	22
2.11.2	The ϵ -Based Multi-Objective Evolutionary Algorithm (ϵ -MOEA) . .	22
2.12	Conclusion	23
3	Multi-Objective Parameter Optimization in Hadoop	27
3.1	Introduction	28
3.2	Background	30
3.2.1	Apache Hadoop	30
3.2.2	Hadoop Configuration Parameters	32
3.2.3	Genetic Algorithms (GA)	32
3.2.4	Generational and Steady-State Genetic Algorithm	33
3.2.5	Steady-State Non-dominated Sorting Genetic Algorithm II	34
3.3	Proposed Optimization System	34
3.3.1	Objective Function	35

3.3.2	System Architecture	36
3.3.3	Parameter Selection	36
3.3.4	System Process Overview	37
3.3.5	Chromosome Representation	40
3.3.6	Fitness Functions	41
3.4	Multi-Objective Hadoop Parameter Optimization Algorithm	41
3.5	Performance Evaluation	43
3.5.1	Hadoop Cluster Configuration	44
3.5.2	Experimental Benchmarks	44
3.5.3	Experimental Analysis	45
3.6	Related Work	49
3.7	Conclusion	50
4	Multi-Objective Optimization of Scientific Workflows in the Cloud	53
4.1	Introduction	53
4.2	Background	57
4.2.1	Scientific Workflows	57
4.2.2	Constrained Optimization	58
4.2.3	Multi-Objective Evolutionary Algorithms (MOEAs)	59
4.3	Proposed Optimization System Model	64
4.3.1	Problem Formulation	64
4.3.2	Decision Variables	65
4.3.3	Mathematical Models for Objective Functions	65
4.4	Optimization System	70
4.4.1	System Architecture	70
4.4.2	Cloud System Model	72
4.4.3	Chromosome Representation	75

4.4.4	Balanced Task Assignment	76
4.4.5	Genetic Operators	78
4.4.6	Penalty Function	79
4.5	Multi-objective Workflow Optimization Algorithm	80
4.6	Performance Evaluation	84
4.6.1	Experimental Scientific Workflow Applications	85
4.6.2	Experimental Setup	86
4.6.3	Experimental Analysis	86
4.7	Related Work and Discussion	96
4.8	Conclusion	99
5	Discussion	106
5.1	Hadoop Configuration Optimization	106
5.2	Scientific Workflow Execution Optimization	108
6	Conclusion and Future Work	118
6.1	Conclusion	118
	Author's Publications	120

List of Figures

1.1	Two Resource Optimization Models for Hadoop and the Cloud.	6
1.2	Thesis Outline	8
2.1	Types of Clouds.	11
2.2	Hadoop Overview.	19
2.3	The Simple Evolutionary Multi-Objective Optimizer Algorithm.	22
2.4	ε -Based Multi-Objective Evolutionary Algorithm.	23
3.1	Data Flow in MapReduce Process	31
3.2	Difference between Generational NSGA-II and ssNSGA-II	34
3.3	Multi-Objective Hadoop configuration optimization system.	36
3.4	Conversion from Binary Chromosome to Real Parameters.	40
3.5	Hadoop Cluster Experimental Environment	44
3.6	Optimization Result for Sort Workload.	47
3.7	Optimization Result for Terasort Workload.	47
3.8	Optimization Result for Wordcount Workload.	48
3.9	Optimization Result for Pagerank Workload.	48
4.1	An Example DAG Workflow.	58
4.2	Process Flow of NSGA-II Optimization.	60
4.3	Non-dominated Sorting in NSGA-II.	62
4.4	Crowding Distance Operation.	62

4.5	Reference Points for Three-Objective Optimization [29].	64
4.6	Proposed Multi-Objective Scientific Workflow Optimization System.	70
4.7	Level-Wise Workflow Partitioning.	72
4.8	Public Cloud System Model.	73
4.9	Chromosome Structure for Optimization during Workflow Execution.	75
4.10	Workflow to Chromosome Mapping.	76
4.11	Level-Wise Balanced Task Assignment.	77
4.12	Crossover Operation.	78
4.13	Mutation Operation.	78
4.14	Structure of the Experimental Scientific Workflows.	85
4.15	2-Objective Cost_Makespan Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large Input Data Size using NSGA-II	88
4.16	Cost_Makespan Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large input data size using NSGA-II	90
4.17	Cost_Failure Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large Input Data Size using NSGA-II	91
4.18	Cost_Makespan Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large Input Data Size using SPEA2	92
4.19	Cost_Failure Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large Input Data Size using SPEA2	93
4.20	Pareto Solutions a, b, and c for Three Workflows with Small, Medium, and Large Input Data Size using NSGA-III	97
5.1	Cost-Makespan Pareto Solutions for Epigenomics Workflow on Private Cloud System.	109
5.2	Cost-Makespan Pareto Solutions for Montage Workflow on Private Cloud System.	109

5.3	Cost-Makespan Pareto Solutions for CyberShake Workflow on Private Cloud System.	110
5.4	Cost-Makespan Pareto Solutions for Epigenomics Workflow on Public Cloud System.	111
5.5	Cost-Makespan Pareto Solutions for Montage Workflow on Public Cloud System.	111
5.6	Cost-Makespan Pareto Solutions for CyberShake Workflow on Public Cloud System.	112
5.7	Cost-Failure Pareto Solutions for Epigenomics Workflow on Public Cloud System.	112
5.8	Cost-Failure Pareto Solutions for Montage Workflow on Public Cloud System.	112
5.9	Cost-Failure Pareto Solutions for CyberShake Workflow on Public Cloud System.	113
5.10	4-Objective, Cost-Makespan Pareto Solutions for Epigenomics Workflow.	113
5.11	4-Objective, Cost-Makespan Pareto Solutions for Montage Workflow.	115
5.12	4-Objective, Cost-Makespan Pareto solutions for CyberShake Workflow.	116

List of Tables

3.1	HDFS and MapReduce parameters	37
3.2	YARN and YARN related MapReduce parameters for x-large instance . . .	38
3.3	YARN and YARN related MapReduce parameters for large instance	38
3.4	YARN and YARN related MapReduce parameters for medium instance . .	39
3.5	YARN and YARN related MapReduce parameters for small instance	39
3.6	Machine Instance Pricing	39
3.7	Benchmark Program used in the Experiment	45
3.8	Single-Objective Hadoop Parameter Optimization Results	46
4.1	Machine Instance CPU Performance	73
4.2	Machine Instance Pricing	74
4.3	Experimental Workflows	85
4.4	Algorithm Parameters	86
4.5	Penalty Values for Constrained Optimization	86
4.6	Convergence, Diversity and Execution Time for SPEA2 and NSGA-II . .	95
5.1	Resource Setting for Epigenomics Workflow (Red Point Solution)	114
5.2	Optimized 4-objective Values for Epigenomics Workflow (Red Point Solution)	114
5.3	Resource Setting for Epigenomics Workflow (Green Point Solution)	114

5.4	Optimized 4-objective Values for Epigenomics Workflow (Green Point Solution)	114
5.5	Resource Setting for Montage Workflow (Red Point Solution)	115
5.6	Optimized 4-Objective Values for Montage Workflow (Red Point Solution)	115
5.7	Resource Setting for Montage Workflow (Green Point Solution)	116
5.8	Optimized 4-Objective Values for Montage Workflow (Green Point Solution)	116
5.9	Resource Setting for CyberShake Workflow (Red Point Solution)	116
5.10	Optimized 4-Objective Values for CyberShake Workflow (Red Point Solution)	117
5.11	Resource Setting for CyberShake Workflow (Green Point Solution)	117
5.12	Optimized 4-Objective Values for CyberShake Workflow (Green Point Solution)	117

Chapter 1

Introduction

The term Big Data refers to enormous amount of structured and unstructured data from several sources that form a key ingredient for many successful organizations. However, a significant portion of Big Data is currently being underutilized or underexploited owing to the overwhelming scale of the data involved. Enterprises are also finding that the use of Big Data is also becoming crucial in their efforts to gain a competitive advantage. According to some surveys, approximately 2.3 trillion gigabytes of data are created every day. The figure is expected to increase to 43 trillion gigabytes by 2020. The cloud provides technology to all consumers in the most cost-effective packages, which is one of the reasons for the data explosion across the web. These high volumes of data present a challenge to the cloud and research is actively being conducted to optimize the processing and analytics of these Big Data more effectively and efficiently.

Cloud computing is the delivery of computing resources and services through the internet. It is rapidly overtaking traditional in-house systems as a reliable, scalable and cost-effective IT solution. Traditional datacenters consist of various pieces of hardware that are connected to a network via a remote server. All users access and share that traditional datacenter and it is very difficult for businesses to upgrade or scale up to support more users. Unlike those traditional systems, the cloud provides for scaling up and down the resources

at any time, which is very persuasive facility for Big Data. Recently, utilization of the cloud for Big Data applications has increased significantly. Moreover, cloud and Big Data together are very effective technologies for enterprises as they can reduce the expenditures of enterprises.

In this research, a resource optimization system was implemented for Big Data applications in cloud computing infrastructure that optimizes service provisioning in terms of cost and performance. This was motivated by the recent trend for widespread use of cloud resources in Big Data experiments that face the challenge of application oriented rightsized resource selection in cloud computing environments. As a result, the system helps with the selection of the right resources during workflow executions that benefits both users with optimized cost and performance and cloud service providers with improved SLA related service provisioning.

1.1 Motivation for this Thesis

Nowadays, the exponential growth in the scientific area makes data analytics difficult. Cloud computing has become one of the best options for high-performance processing although several high-performance computing environments such as cluster computing, grid computing and cloud computing, exists, because scientists can scale up and down their infrastructure resources according to the application requirements and budget in the cloud. Furthermore, the cloud provides computing as a utility service in the form of Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In the context of cloud computing, an application's execution cost is simply the monetary cost of renting resources from the cloud service providers. There are two possible issues: resource over-provisioning and resource under-provisioning if the rented resources do not match with the application needs. From the users' perspective, they want to execute their applications as fast as possible at the lowest cost. Addressing the issue of minimizing re-

source usage with the best performance in the cloud is a challenging research issue.

Our proposed optimization system deals with the search for optimal resources for Big Data applications in Hadoop and cloud infrastructure during experimentations. In this scenario, the Hadoop cluster is implemented using the virtual machine(VM) instances of the cloud infrastructure and the monetary cost for application execution in the cloud is based on the amount of resources used, i.e., task execution cost correlated with the total number of CPU cycles for all tasks. By using cloud infrastructure, customers can also reduce the processing cost of workflow applications with the aid of the cloud's pay-per-use model. Moreover, a task that shares the same time interval with a previous task hosted in the same instance might not result in extra cost, reducing the overall workflow execution cost.

1.2 Objectives of this Thesis

The thesis focuses on optimization of Big Data applications by means of resource optimization through multi-objective evolutionary algorithms (MOEAs). The major objectives are as follows:

- Development of a resource optimization system that gives tangible and intangible benefits to both stakeholders (users and service providers) in service provisioning environments.
- Speed up scientific research experimentation in computing infrastructure by means of current popular technologies such as Hadoop and the cloud.
- Proposal of an approach that can solve the challenge of rightsized resource provisioning in cloud computing infrastructure that provides optimized cost performance tradeoff during service execution, to better maintain cloud's Service Level Agreement (SLA).

1.3 Contributions of this Thesis

This thesis conducted resource optimization in two kinds of infrastructure; Hadoop and the cloud. The first portion of this research focuses on configuration optimization and set up cost optimization of input workload applications on the Hadoop platform using cloud VM instances. The contribution made towards Hadoop optimization is identification of sensitive parameters from Hadoop configuration files and searching optimized configuration plans for the input application using a multi-objective evolutionary algorithm (MOEA) called steady-state Non-dominated Sorting Genetic Algorithm (ssNSGA-II).

The latter portion of this research focuses on workflow execution in the cloud infrastructure. Most of the previous research studies on scientific workflow applications focused on grid infrastructure. However, grid computing is being supplanted by cloud computing owing to advantages such as pay-per-use and elastic resource facilities. Consequently, further optimization research on cloud infrastructure has become a necessity in workflow optimization. In our approach, we conduct multi-objective workflow optimization, i.e., minimization of workflow makespan, VM instance resource cost and failure probability during workflow execution, with maximization of VM availability probability during resource deployment, without violating the task dependencies of the workflow in the cloud infrastructure. System optimization model of our proposed approach also conducts load balancing of task assignment to the VMs during processing. Thesis contributions are as follows:

- Modeling the resource optimization problem as the (constrained) multi-objective optimization problem
- Level-wise optimization for task dependency, resource sharing and elasticity
- Randomized search-based resource optimization using a MOEA for better cloud service provisioning

- Balanced task clustering and assignment for better load balancing to improve system reliability through minimization of failures and maximization of availability.

1.4 Problem Origin of the System

This research work stems from the two challenges. Firstly, nowadays, much research is being conducted on Big Data analytics with the Hadoop MapReduce framework. However, Hadoop parameter configuration is very challenging because of the complexity of its framework. Configuring Hadoop parameters to minimize the execution time of MapReduce jobs in Hadoop cluster still requires much research. Further, determining the type of machine instances that should be used to minimize the resource usage cost for executing applications in clusters is also difficult. In this case, the parameter optimization methodology that addresses the above problems in Hadoop clusters is a challenging research area.

Secondly, cloud computing is a popular alternative for processing large-scale data in scientific experiments owing to its resource elasticity and easy-to-use paradigms. According to previous surveys conducted by RightScale (2017), the major usage of cloud computing is in scientific experimentations and the enterprise public cloud adoption has increased significantly during recent years. However, cloud still has several challenges, and lack of expertise in using the cloud is cited by 25 % of cloud users as a major concern. Thus, resource optimization that gives the best performance and the lowest financial deployment cost is a challenging issue preventing increased cloud adoption in several domain areas.

1.5 Overview of Multi-Objective Resource Optimization of Big Data Applications

Resource optimization for Hadoop clusters and the cloud infrastructure are very important in today's Big Data processing application environment. Our proposed models are

Hadoop Resource Optimization Model
(through parameter tuning)



Cloud Resource Optimization Model for
Scientific Workflows

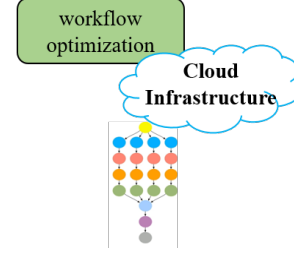


Figure 1.1: Two Resource Optimization Models for Hadoop and the Cloud.

proposed to solve the problem of optimizing resources of Big Data applications in Hadoop and the cloud infrastructure using the optimization technology. Users need not manage the resources for the application on their own and the two proposed approaches conduct application oriented resource optimization providing processing efficiency and cost effectiveness.

Resource optimization was conducted using the multi-objective evolutionary algorithms. There are two approaches proposed in our resource optimization system illustrated in Figure 1.1: Hadoop configuration optimization and scientific workflow optimization. The experiments on Hadoop configuration optimization were conducted using the actual high specification Hadoop clusters whereas workflow optimization model experiments were conducted on a simulated cloud system environment.

This thesis combines cloud computing, Big Data and optimization technologies. The recent trend towards the usage of cloud in Big Data applications raises the challenge of which resources should be used for the input application. The cloud's mandatory facility to decide the resource requirements of the application implies extra work rather than focusing on their subjective experiments. Moreover, it is also difficult for the cloud service providers to provide services without violating SLA. Tackling those problems is motivated

by researchers who try to combine the benefits of cloud, big data and optimization to bring them closer together. This thesis proposes solutions for the resource optimization problem of Big Data applications and combines the good features of the recent technology trends Big Data and the cloud with optimization technology.

1.6 Benefits of the System

The proposed research work enables better cloud resource deployments for running Big Data applications and improves the current scientific research experiments to be more convenient, effective and efficient for scientists during experiments. Consequently, the experimental results from effective scientific research will bring invaluable benefits to the society.

1.7 Organization of this Thesis

This thesis is organized into six chapters as shown in Figure 1.2, with the core chapters comprising several symposia, conferences, and journal papers published during the Ph.D. course.

First, Chapter 1 presents general information of this thesis, the motivations of this thesis, contributions, and the objectives of this thesis. It also highlights the current problem situation and the reasons why we chose to do our research in this field of study. The possible benefits of our research to the society are also presented. Chapter 2 contains background and literature reviews of the related technologies of the thesis. Chapter 3 presents detailed explanation of multi-objective optimization of Big Data Applications in Hadoop cluster environments. The chapter includes the reasons for the growing use of Hadoop in Big Data processing applications and then describes the importance of parameter optimization in the Hadoop infrastructure. Details of the optimization process using steady-

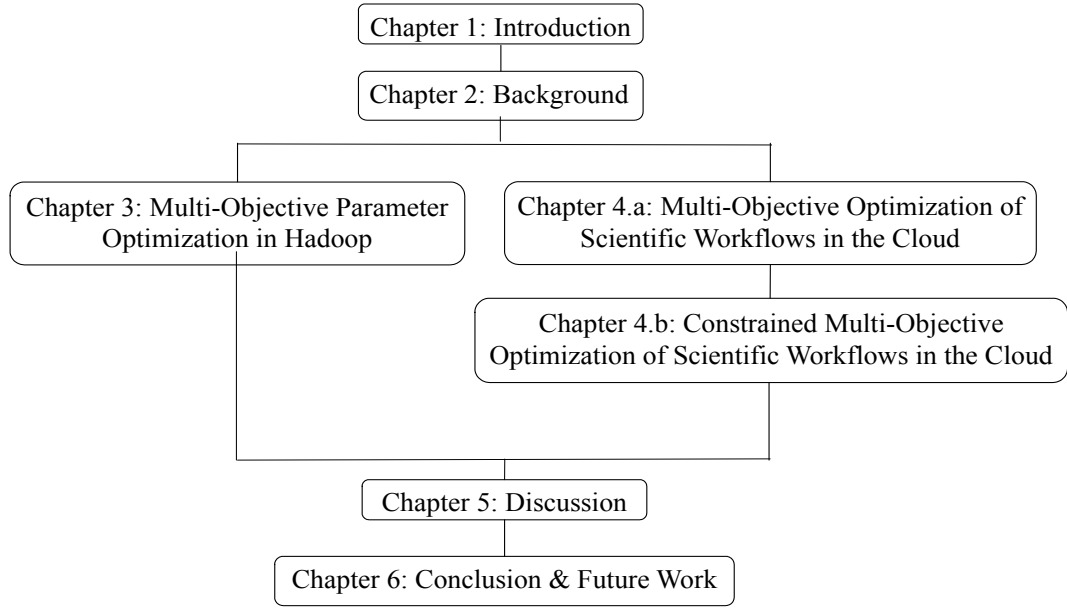


Figure 1.2: Thesis Outline

state Non-Dominated Sorting Genetic Algorithm II (ssNSGA-II) are also described with relevant experimental results from real Hadoop clusters. Chapter 4 presents the proposed multi-objective optimization of big scientific applications in the private cloud system model and public cloud system model, along with implementation details and evaluations of the proposed system. Performance comparison with the Strength Pareto Evolutionary Algorithm (SPEA2) and Non-Dominated Sorting Algorithm II (NSGA-II) for the optimization process is also presented to identify the most suitable algorithm for our approach. Further, multi-objective optimization of Big Data applications as a constrained optimization problem is proposed and model effectiveness evaluation results are presented. The system model verification was conducted via simulation and Python programming language was used to implement the system. Chapter 5 discusses our research findings (advantages and limitations) of current optimization approaches in Hadoop and in the cloud Infrastructure and potential solutions for the limitations of the approaches. Chapter 6 concludes this thesis with a summary of the work conducted and suggestions for future research.

Chapter 2

Background

This chapter introduces the context of the popular technologies such as cloud computing and Big Data which are the motivations for doing this research, together with some related background theory concerning optimization and scientific workflows. It starts with the general overview of the cloud, its features, deployment models and a layered architecture of the cloud. Then, the characteristics of Big Data are presented, followed by an outline of the nature of Hadoop and Scientific Workflows, and an overview of multi-objective optimization algorithms (MOEAs) and their key concepts.

2.1 Cloud Computing

Cloud computing presents a dynamic environment in which resources are provisioned on demand. Cloud-based technology constitute services provided through the internet. Rather than having the content or the software installed on an on-premise computer or any other computing device, one can have it on a network that can be accessed from any part of the world through a computing device and the internet. There are various benefits of the cloud services that are also considered by many areas and in educational institutions as well for experimentations and data analysis reasons. It is a scalable distributed computing envi-

ronment in which a large set of virtualized computing resources, different infrastructures, various development platforms, and useful softwares are delivered as a service to customers as in a pay-as-you-go manner. It is a paradigm shift from mainframe to client-server that preceded it in the early 1980s. Details are abstracted from the users who no longer need to have expertise in, or control over the technology infrastructure “in the cloud” that supports them.

NIST [14] defines cloud computing as follows:

“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. ”

Cloud computing brings efficiencies and savings. The diverse benefits of cloud computing are undoubtedly worth pursuing. Cost-cutting is at the top of most companies’ lists of priorities in these challenging economic times. Having turned from revolutionary possibility into increasingly well-established custom, the cost of “outsourcing to the cloud” is now falling dramatically. In only paying for the resources used, therefore, operating costs can be reduced. After all, in-house data centers typically leave 85%-90% of available capacity idle. Cloud computing can also lead to energy savings, removing from individual companies the costly burden of running a datacenter plus generator back-up and uninterruptible power supplies. Thus, it results in reduction of capital expense and operation expense.

2.2 Features of Cloud Computing

Cloud computing brings many new features into the computing paradigms. These features are briefly described below [1].

- Scalability and on-demand Services: Cloud computing provides resources and ser-

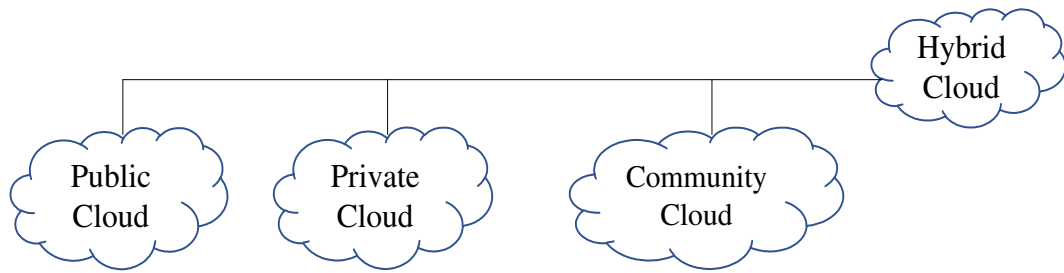


Figure 2.1: Types of Clouds.

vices for users on demand. The resource is scalable over several data centers.

- **User-centric Interface:** Cloud interfaces are location independent and can be accessed via well-established interfaces such as Web services and Internet browsers.
- **Guaranteed Quality of Service (QoS):** Cloud computing can guarantee QoS for users in terms of hardware or CPU performance, bandwidth, and memory capacity.
- **Autonomous System:** Cloud computing systems are autonomous systems managed transparently to users. However, software and data inside clouds can be automatically reconfigured and consolidated to a simple platform depending on user's needs.
- **Pricing:** Cloud computing does not require up-front investment. No capital expenditure is required. Users pay for services and capacity as they need them.

2.3 Cloud Deployment Models

There are four main types of cloud deployment models: Public cloud, Private cloud, Community cloud and Hybrid cloud as shown in Figure. 2.1.

2.3.1 Public Cloud

In simple terms, public cloud services are characterized as being available to clients from a third-party service provider via the Internet. The term “public” does not always

mean free, even though it can be free or inexpensive to use. A public cloud does not mean that a user's data is publicly visible; public cloud vendors typically provide an access control mechanism for their users. Public clouds provide an elastic, cost effective means to deploy solutions [15].

2.3.2 Private Cloud

Private clouds are built for the exclusive use of one client, providing the utmost control over data, security, and QoS. The company owns the infrastructure and has control over how applications are deployed on it. Private clouds may be deployed in an enterprise datacenter, and they also may be deployed at a co-location facility. They can be built and managed by a company's own IT organization or by a cloud provider [12]. In this "hosted private" model, a company such as Sun can install, configure, and operate the infrastructure to support a private cloud within a company's enterprise datacenter. This model gives companies a high level of control over the use of cloud resources while bringing in the expertise needed to establish and operate the environment.

2.3.3 Community Cloud

Community clouds are controlled and used by a group of organizations that have shared interests, such as specific security requirements or a common mission [9]. The idea of a community cloud is derived from the grid computing and volunteer computing paradigms. In a community cloud, several enterprises with similar requirement can share their infrastructures, thus increasing their scale while sharing the cost. Another form of community cloud may be established by creating a virtual data center from VM instances deployed on underutilized users' machines.

2.3.4 Hybrid Cloud

Hybrid clouds combine public and private clouds that interoperate [9]. These clouds would typically be created by the enterprise, and management responsibilities would be split between the enterprise and public cloud provider. The hybrid cloud leverages services that are in both the public and private space. Hybrid clouds can provide the necessary services when a company needs to employ the services of both a public and private cloud. In this model users typically outsource no business-critical information and processing to the public cloud, while keeping business-critical services and data in their control.

2.4 Cloud Service Layers

Cloud service providers tend to offer services that can be grouped into three categories: software as a service, platform as a service, and infrastructure as a service.

2.4.1 Infrastructure as a Service (IaaS)

Although most existing cloud computing implementations share the common high level notion of flexible, scalable, and dynamic computational “provisioning”, there is significant variation in exactly how that power is presented to the end user. Some systems, such as Amazon’s Elastic Compute Cloud (EC2) and Enomolism, allow users to allocate entire VMs on demand, thus providing what is commonly referred to as IaaS. Here, the users are responsible for providing the kernel, base Operating System (OS) software, and any user level software and applications they wish to run and the IaaS system provisions physical resources and instantiates the user’s VMs [5].

Specifically, commercial cloud infrastructures take advantage of the ability to control the local resource configuration (hardware versioning, OS versioning, network and storage policies, etc.) and access to large collections of potentially expensive resources (e.g., publicly visible and routable internet addresses). In a research setting, it is not unlikely that

the cloud infrastructure can mandate a specific configuration for all hardware and software it manages, nor is it possible to predicate functionality on the availability of very large resource sets [5].

Further, IaaS systems typically target a specific installation; they are not engineered with extensibility or portability as a primary concern, nor is the need for ease of system administration given primacy in the design. The difficulties are compounded by the need to be able to incorporate multiple compute clusters into a single resource pool from which cloud allocations are to be drawn. Few open-source software packages of any kind are designed to be installed and deployed on multiple compute clusters that operate together as an ensemble.

2.4.2 Platform as a Service (PaaS)

PaaS is the delivery of a computing platform, often an OS with associated services, over the internet without downloads or installation. PaaS systems are typically development platforms designed to operate specifically in the cloud environment. G Suite [18], Google Apps is an example of PaaS. PaaS implementations typically have the following [4]:

- Integrated Development Environment Services: there are many variation of PaaS offerings with the most all-inclusive having an integrated development environment with all the necessary source code, version controls, and audit capabilities.
- Interface Creation Tools
- Web and Database Integration

2.4.3 Software as a Service (SaaS)

SaaS is the delivery of a licensed application to customers over the internet for use as a service on demand. A SaaS vendor may host an application and allow the customer to

download the application for a period, after which the application becomes inactive. SaaS architectures are classified into one of four levels [4]:

- Custom: At this level, each system user has his or her own customized version of the application that runs on the host servers.
- Configurable: At this level, users can separate instances out of the same application rather than being constrained to just their own version of the application.
- Configurable, Multitenant-efficient: This builds on the previous level by allowing a single application to serve all users.
- Scalable, Configurable, and Multitenant-efficient: This level adds scalability to the previous level allowing for load balancing and the increasing or decreasing of system capacity based on user demand.

2.5 Big Data

Big Data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze [8]. Big Data management is the capability to manage a huge volume of disparate data, at the right speed, and within the right time frame to allow real-time analysis and reaction [7]. It is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools [2]. The characteristics of Big Data includes the following [17]:

1. Volume

The volume of data is the major challenge in Big Data that concerns with storing gigabytes or petabytes of data which will be continuously growing in the future.

2. Variety

Variety refers to the different formats of data such as unstructured, structured, and

semi-structured data that relational database system are not able to handle. The data sources of Big Data come in several forms such as documents, emails, social media text messages, audio, video, graphics, sensor data, machine logs, and DNA analysis data.

3. Velocity

Big Data not only include data-at-rest but also stream data which is data-in-motion. In that case, the major concern is the consistency and completeness of fast pace streams of data and getting the desired result matching. The processing of these data is a challenging issue in the current information technology area.

4. Veracity

Veracity describes the quality of data; accuracy and completeness of the data is also concerned.

The above characteristics make Big Data difficult to process on a single system that highlights the criticality of high performance computational infrastructure. Digital media, pervasive sensing, web authoring, mobile computing, scientific and medical instruments, physical simulations and virtual worlds are all delivering vast new datasets relating to every aspect of our lives. Although there are several sources of data, this dissertation focuses on Big Data coming from scientific experiments such as bioinformatics, and environmental, astronomic field of areas.

2.6 Challenges of Big Data

Society is becoming increasingly more technology and information driven, and thus, organizations are producing and storing vast amounts of data. Data management to retrieve the insights from the produced Big Data is a challenge and key to competitive advantage. Although Big Data analytics enables substantial benefits to the community (businesses,

researches, organizations etc.), there are still challenges in conducting effective data analytics. Big data challenges include the following [10]:

1. **Data Variety:** When the data are unstructured, it is difficult to extract meaningful content from those data. Thus, data variety becomes one of the open challenges of Big data processing.
2. **Data Storage:** The storage management of unstructured data and the extracted insights is also another important open big data challenge. The growing rate of Big Data requires development of a better strategy for efficient data retrieval.
3. **Data Integration:** In Big Data processing environment, it is necessary to manage data of different nature which results in the challenge of how to integrate those data (structured, unstructured and semi-structured).
4. **Data Processing and Resource Management:** Resource management of computing infrastructure in processing Big Data applications and analytics is a very big problem although new programming models are developed to better support Big Data processing and analytics.

2.7 Cloud Computing and Big Data

Two Big Data IT initiatives: cloud computing and Big Data are very popular terms for businesses, organizations, and research institutes. The aim of cloud computing is to deliver computing as a solution for managing Big Data that comprise structured, semi-structured, and unstructured information. The cloud provides services on a pay-per-use model, and has the potential and capability to enhance business awareness and productivity by enabling greater efficiencies and reducing costs of processing Big Data applications. Further, most of the data required for Big Data applications reside in several data sources, some of which are already resident in the cloud. Thus, processing the applications where the data

reside enables more efficient data processing.

However, the cloud provides a customized resource selection facility to users and that makes it difficult to decide on application oriented optimal resources. Resource management of Big Data processing is one of the critical issues in cloud and Big Data.

2.8 Hadoop and Big Data

Hadoop is an open source Java-based framework for handling large sets of data in distributed environment [3]. It is designed for scaling up from a single server to thousands of servers in a cluster with local computation and storage on every individual server. There are three building blocks in Hadoop: HDFS, YARN and MapReduce. HDFS stands for Hadoop distributed file system and it is responsible for data storage inside the Hadoop cluster. YARN (Yet Another Resource Negotiator) is responsible for providing computational resources such as CPUs, memory, I/O, network etc. MapReduce performs data processing in a parallel fashion by mapping data into a collection of [key,value] pairs and reducing those [key,value] pairs with the same key. Hadoop MapReduce enables efficient handling of structured, unstructured and semi-structured data that makes Hadoop suitable for big data processing and analytics. However, Hadoop provides customized configuration parameters for the cluster resources such as CPU, memory, I/O, network, etc. Although Hadoop is a good alternative to run big data applications, it is necessary to adjust the cluster resources through Hadoop configuration parameters. Figure 2.2 depicts the overview of Hadoop. Further details of Hadoop MapReduce are described in Chapter 3.

2.9 Scientific Workflows

The term workflow initiated from a business process consisting of one or more linked procedures or activities that collectively realize a business objective or policy goal, within

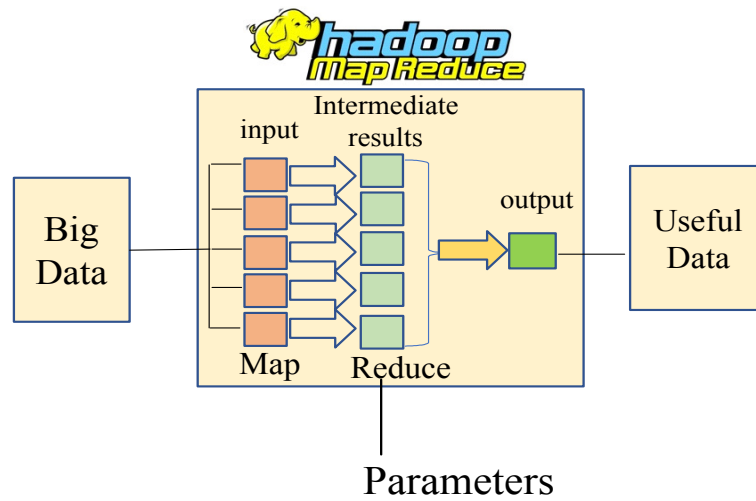


Figure 2.2: Hadoop Overview.

the context of an organizational structure, defining functional roles and relationships. It is defined as “The automation of a business process, in whole or parts, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules” [16]. The term workflow gradually entered several fields concerned with handling a series of tasks with dependencies. The current popular workflow area is in the science area. Researchers are currently conducting data analytic experiments by using scientific workflows.

A scientific workflow is a flow of tasks, mostly complex computational tasks, that are part of a scientific experiment. These workflows are executed on distributed systems because they require high computation power. There are four phases in a scientific workflow life cycle [16]:

1. Composition: Composition allows a user to specify the phases and dependencies among tasks in an abstract manner. Directed Acyclic Graph (DAG) with tasks and inter-job dependencies are represented in XML (DAX) a popular authoring language for workflow composition.
2. Planning: Planning maps workflow tasks to computational resources for execution. It transforms a given workflow to an executable workflow.

3. Execution: This phase schedules the workflow jobs to the execution resources based on the planned information in the workflow. Resources are provided through the cloud infrastructure. The resource selection is also conducted in this phase if it was not done in the planning phase. During execution, the workflow status and its jobs are continuously monitored to detect the job completion or failure and the output is retrieved if the job is successful.
4. Provenance: Provenance involves capturing the associated provenance for the workflow's output. The provenance information includes job execution environment, parameters being used or input data provided to the job, job logs, intermediary outputs and the final output produced during the execution of the workflow.

2.10 Cloud Computing and Scientific Workflows

Our research is concerned with workflow resource management and execution, which are related with the planning and execution phases of the workflow. Previous studies conducted scheduling the workflow tasks to reduce the makespan of scientific workflow applications on several computing infrastructures, such as computing clusters and the grids. Resource provisioning and management represents the functionality and mechanism of allocating computing resource, storage space, network bandwidth, etc., to specific workflows [18]. Nowadays, the data growth rate of scientific experiments is quite rapid and it is very difficult to execute the scientific workflow applications in the traditional workflow management systems. The adoption of cloud computing to process these scientific data has increased recently because the cloud provides facilities such as easy setup, resource elasticity, and a pay-per-use model. However, the major goal of scientific workflow is enabling scientists to focus on domain-specific (science) aspects of their work, rather than dealing with complex data management and software issues. Although the cloud provides infrastructure service for scientific workflow applications as IaaS, the user needs to choose the

required resources for their applications. Consequently, the resource provisioning problem has become a challenging research area involving execution of the scientific workflow applications in the cloud. This dissertation focuses on solving the resource provisioning problem in the cloud using a multi-objective evolutionary optimization (MOEA) approach.

2.11 Multi-Objective Evolutionary Optimization (MOEA)

MOEAs have been widely used in real-world applications such as engineering, design optimization. A multi-objective optimization problem (MOP) can be stated as follows [6]:

$$\begin{aligned} &\text{Minimize } F(x) = (f_1(X), f_2(X), \dots, f_m(X)) \\ &\text{Subject to } x \in \Omega \end{aligned} \tag{2.1}$$

where, the decision vector $x = (x_1, x_2, \dots, x_n)$ belongs to the (nonempty) decision space Ω , the objective function vector $F : \Omega \rightarrow \Lambda$ consists of $m(m \geq 2)$ objectives, and the objective space Λ is usually equal to $S \subset R^m$ where, m = number of objectives. If m is larger than 3, the problem becomes a many-objective optimization problem. The goal of multi-objective optimization is to approximate the Pareto Front (PF) in the objective space so that no further improvement on any objective can be achieved without harming the other objectives.

Evolutionary Algorithms are population-based, black-box search/optimization methods and they do not need assumptions like continuity or differentiability. They are very suitable for dealing with MOPs. Numerous studies have been done on MOEAs and researchers have contributed many algorithms. An overview of some popular Pareto-Based EAs is given in the following section. Among them, we applied SPEA-2, NSGA-II and NSGA-III in our research and thus further details of those algorithms are presented in Chapters 3 and 4.

2.11.1 The Simple Evolutionary Multi-Objective Optimizer (SEMO)

SEMO [11] represents the simplest population based EA and was originally proposed to conduct binary optimization by Laumanns et al. in 2003. However, it is easily extendable to continuous optimization. Like other optimization algorithms, it begins with a random initial population with a size of one. The initial population size grows as the algorithm finds non-dominated solutions during optimization. The offspring is created by mutating one of the individuals in the population at each generation and the population is update if the child generated dominates one of the individuals in the population [13]. Figure 2.3 indicates how SEMO works during optimization.

1. Choose an initial individual x uniformly at random from the set X
2. $w(x) := 0$
3. Let $P := \{x\}$
4. Loop
5. Select x from $y \in P \mid w(y) \leq w(z), \forall z \in P$ uniformly at random
6. $w(x) := w(x) + 1$
7. Create an offspring $\hat{x}$ by flipping a randomly chosen bit in x
8. $P := P - \{z \in P \mid \hat{x} \succ z\}$
9. If there is no $z \in P$ s.t. $(z \succ \hat{x} \vee f(z) = f(\hat{x}))$ then $P := P \cup \{\hat{x}\}$
10. end loop

Figure 2.3: The Simple Evolutionary Multi-Objective Optimizer Algorithm.

2.11.2 The ε -Based Multi-Objective Evolutionary Algorithm (ε -MOEA)

ε -MOEA [6] was proposed by Deb et al., in 2005. It uses the concept of ε -dominance in a similar manner to the ε -dominance that diversity evolutionary multi-objective optimizer (DEMO) and includes a population and an archive. At each generation, one individual from the population and one from the archive are selected and generate an offspring. If the offspring dominates an individual or individuals in the population, replacement occurs with one or randomly-selected individual to the population. Next, the child is compared

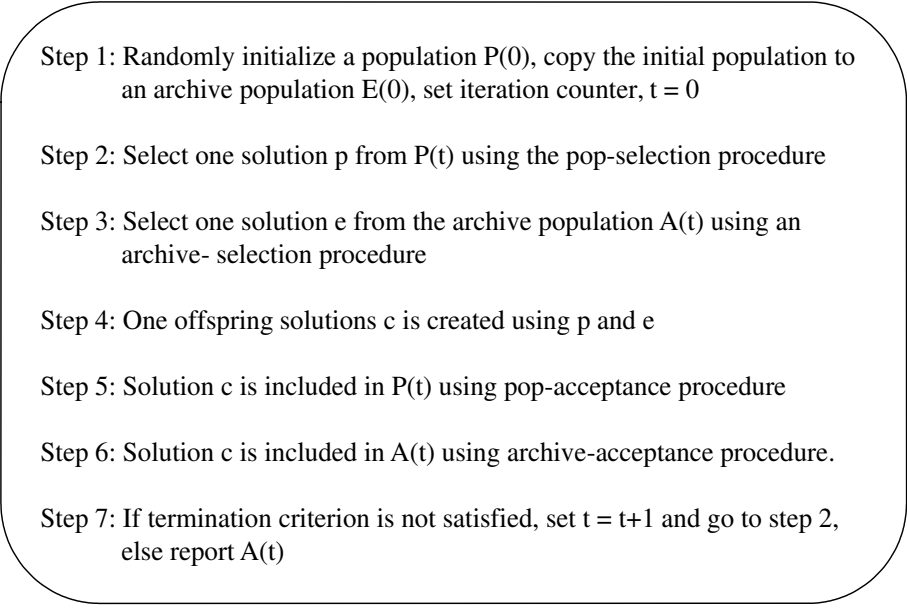
- 
- Step 1: Randomly initialize a population $P(0)$, copy the initial population to an archive population $E(0)$, set iteration counter, $t = 0$
 - Step 2: Select one solution p from $P(t)$ using the pop-selection procedure
 - Step 3: Select one solution e from the archive population $A(t)$ using an archive-selection procedure
 - Step 4: One offspring solutions c is created using p and e
 - Step 5: Solution c is included in $P(t)$ using pop-acceptance procedure
 - Step 6: Solution c is included in $A(t)$ using archive-acceptance procedure.
 - Step 7: If termination criterion is not satisfied, set $t = t+1$ and go to step 2, else report $A(t)$

Figure 2.4: ε -Based Multi-Objective Evolutionary Algorithm.

with the archive for domination that comes with the decision to place in the archive or not. If neither of the above conditions are satisfied, the algorithm calculates the ε -box $B(x)$ of the child x . Then the child and archive comparison is conducted again for the archive update process. The algorithm for ε -MOEA is illustrated in Figure 2.4.

2.12 Conclusion

To the best of my knowledge, there are several issues to be solved on cloud computing and big data processing, data analytics, resource management, data security, etc. In this study, we focused on resource management and optimization of big data applications in Hadoop and the cloud by using some popular multi-objective evolutionary algorithms. The details of our work are presented in the ensuing chapters.

Bibliography

- [1] B. Furht and A. Escalante Editors. *Handbook of Cloud Computing*. ISBN 978-1-4419-6523-3, USA, Springer Press, 2010.
- [2] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li. *Big Data Processing in Cloud Computing Environments* International Symposium on Pervasive Systems, Algorithms and Networks.
- [3] C. Zaveri. *Use of Big-Data in healthcare and lifescience using Hadoop technologies* 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), Feb, 2017.
- [4] D. Barrett and G. Kipper. *Virtualization and Forensics: A Digital Forensic Investigator's Guide to Virtual Environments*. Syngress, Aug 6, 2010.
- [5] D. Nurmi, R. Wolski, C. Grzegorzczak, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. *Eucalyptus: an open-source cloud computing infrastructure*. Journal of Physics: Conference Series 180 (2009) 012051, doi: 10.1088/1742-6596/180/1/012051, 2009.
- [6] D. Simon. *Evolutionary Optimization Algorithms* USA, Wiley Press, 2013.
- [7] J. Hurwitz, A. Nugent, F. Halper, and M. Kaufman. *Big Data for Dummies* Wiley brand, 2013.

- [8] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh and A. H. Byers. *Big Data: The next frontier for innovation, competition, and productivity* McKinsey Global Institute, 2011.
- [9] M. B. Geer and Jr. K. L. Jackson. *Practical Cloud Security: A Cross-Industry View*. CRC Press, Aug 5, 2016.
- [10] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya. *Big Data computing and clouds: Trends and future directions* Journal of Parallel and Distributed Computing, Elsevier, 2015.
- [11] M. Laumanns, L. Thiele, and E. Zitzler. *Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions*. IEEE Transactions on evolutionary computation, vol.i, no.2, 2004.
- [12] M. M. Hassan and E. Huh. *Dynamic Cloud Collaboration Platform: A Market-Oriented Approach*. Springer Science & Business Media, Sep 13, 2012.
- [13] N. Hebbinghaus. *Evolutionary algorithms*. Lecture, 2006.
- [14] P. Mell and T. Grance. *The NIST definition of cloud computing*. National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech., 2011.
- [15] P. R. Baker, D. J. Benny. *The Complete Guide to Physical Security*. CRC Press, Apr 19, 2016.
- [16] S. Bendoukha. *Multi-Agent Approach for Managing Workflows in an Inter-Cloud Environment* Dissertation, Hamburg, 2016.
- [17] S. Sharma and V. Mangat. *Technology and trends to handle big data: survey*. 2015 Fifth International Conference on Advanced Computing and Communication Technologies, IEEE, 2015.

[18] G. Suite

<https://gsuite.google.com/>. Accessed November, 2017.

Chapter 3

Multi-Objective Parameter Optimization in Hadoop

This chapter presents the details of Hadoop Configuration Parameter Optimization to minimize the execution time of MapReduce jobs for Big Data processing and analytics. Hadoop framework becomes a popular processing framework for Big Data applications because of facilities such as scalability, cost effectiveness, flexibility, processing speed and resilient to failures. However, the framework has over hundreds of configuration parameters that have significant effects on the performance of Hadoop cluster. Our proposed Hadoop parameter optimization addressed parameter optimization problem on two real Hadoop clusters. First, single objective Hadoop parameter optimization was conducted on 24 node Hadoop cluster (specifically, one NameNode and 23 Datanodes) using virtual machine (VM) instances from Hokkaido University Academic Cloud System. Second, bi-objective optimization of Hadoop parameter optimization was conducted on high specification Hadoop cluster (one NameNode and 5 DataNodes); each node had 40 cores CPU, 128 GB RAM and 4TB storage. Experiments were conducted on these real Hadoop clusters using steady-state Genetic Algorithm (GA) and steady-state Non-dominated Genetic Algorithm II (ssNSGA-II) for single and bi-objective Hadoop parameter optimizations.

3.1 Introduction

The digital universe is expanding with data at 40% per year and is projected to continue this trend into the next decade, as a result of widespread adoption of computing devices in our everyday lives. The ever-increasing use of connected devices in view of the internet of things is resulting in an exponential increase in Big Data. The major sources of these Big Data include public web, social media, business applications, mobile and sensor applications, machine log data and scientific and simulation data. However, a significant portion of this big data is currently being underutilized or underexploited owing to the overwhelming scale of the data involved. Effective sharing, understanding, and use of this new wealth of raw information pose one of the great challenges of the new century. Enterprises are also finding that the use of Big Data is also becoming crucial in efforts to gain a competitive advantage. In order to get useful information from Big Data within a short period of time, scientists and engineers are adopting new parallel programming models that can facilitate not only data scalability but also automatic resource management.

MapReduce [5] is the most popular parallel programming framework for high performance computing and Big Data analytics. The MapReduce framework comprises two phases: the map phase and the reduce phase. During the map phase, the input job is first divided and assigned to worker nodes by the master node. Then, the map functions in worker nodes convert the received data to equivalent key-value pairs and perform the map operations. In the reduce phase, the master node collects the output of the map functions from the worker nodes and assigns reduce tasks to perform user-defined operations on these key-value pairs by aggregating, combining, filtering, and transforming functions.

Various studies have shown that Hadoop cluster configurations play a significant role in its processing performance [3], [6], [8]. However, Hadoop provides hundreds of configuration parameters and it is very difficult to identify the parameters that can impact the performance of a particular application. Further, utilizing the same configuration setting

for all kinds of MapReduce jobs does not result in optimum across-the-board performance. Thus, configuration optimization of the Hadoop framework for efficient Big Data processing still remains challenging.

The proposed Hadoop parameter optimization started with single objective Hadoop parameter optimization in terms of MapReduce application execution time minimization using a steady-state genetic algorithm followed by bi-objective parameter optimization of Hadoop cluster that optimizes MapReduce application processing in terms of time and cluster setup cost using VM instances from the cloud infrastructure. We chose the steady-state Genetic Algorithm (GA) approach as it iteratively searches for the fittest parameter combination and outputs an optimized configuration parameter list for a particular type of application. Moreover, GA conducts an aggressive search that results in near-optimal configuration. However, for bi-objective optimization, an evolutionary approach was considered because configuration optimization is quite complex and several parameters need to be simultaneously considered during the optimization process to obtain near-optimal solutions. Although only a single criterion, execution time is used in single objective Hadoop configuration optimization. Bi-objective optimization identifies optimized configuration parameters with two criteria: execution time of workload and instance cluster setup cost to execute the workload in the cluster. The experimental results obtained verify the efficacy of our approach. Moreover, bi-objective optimization experimental results show that, the execution time of MR jobs depend not only on the type of machine instance but on the configuration setting. They also highlight that medium and large instance types can give a balance between cluster setup cost and execution time for the current experimental workloads whereas large and small instance types have certain cost and execution time tradeoffs.

The remainder of this chapter is organized as follows. Section 3.2 presents relevant background information. Section 3.3 gives an overview of the proposed system, including its processing steps and implementation. Section 3.4 describes the proposed Hadoop

parameter configuration algorithm. Section 3.5 presents and analyzes the results of the performance evaluation conducted using HiBench workloads on a six-node Hadoop cluster. Section 3.6 discusses related work. Finally, Section 3.7 presents concluding remarks and outlines possible future work.

3.2 Background

This section describes the background related to the Hadoop configuration optimization system by explaining the two major theories, Apache Hadoop and Steady-State Non dominated Sorting Genetic Algorithm (ssNSGA-II), applied in the proposed system model.

3.2.1 Apache Hadoop

The pace at which data accumulate necessitates better algorithms that facilitate analysis and produce actionable results. In addition, there is the added question of how to store and analyze these data effectively. Hadoop is an open source Apache software framework that can evaluate gigabytes or petabytes of structured or unstructured data and transform them into a more manageable form for applications to process. MapReduce is a linearly scalable programming model for processing large-scale data. It performs processing using map and reduce functions, each of which defines a mapping from one set of key-value pairs to another. These functions are also scalable, which means that they can be used unchanged on both small and massive datasets. Data access in MapReduce is fast as it provides data locality. Further, it operates at high level, which enables programmers to fully focus on the behavior of the system. In the map stage, data elements from a list of such elements are inputted one at a time to a Mapper that transforms each element individually into an output data element. In reduce phase, the output map values are aggregated and a single final output value is returned. The map and reduce tasks are scheduled using YARN (Yet Another Resource Negotiator) and run on nodes in the cluster. If a task fails, it will be

automatically rescheduled to run on a different node. When executing applications, Hadoop divides the input to a MapReduce job into fixed-size pieces called splits where each split corresponds to a single map task. Figure 3.1 shows the simple MapReduce data flow with multiple reduce tasks as in [14].

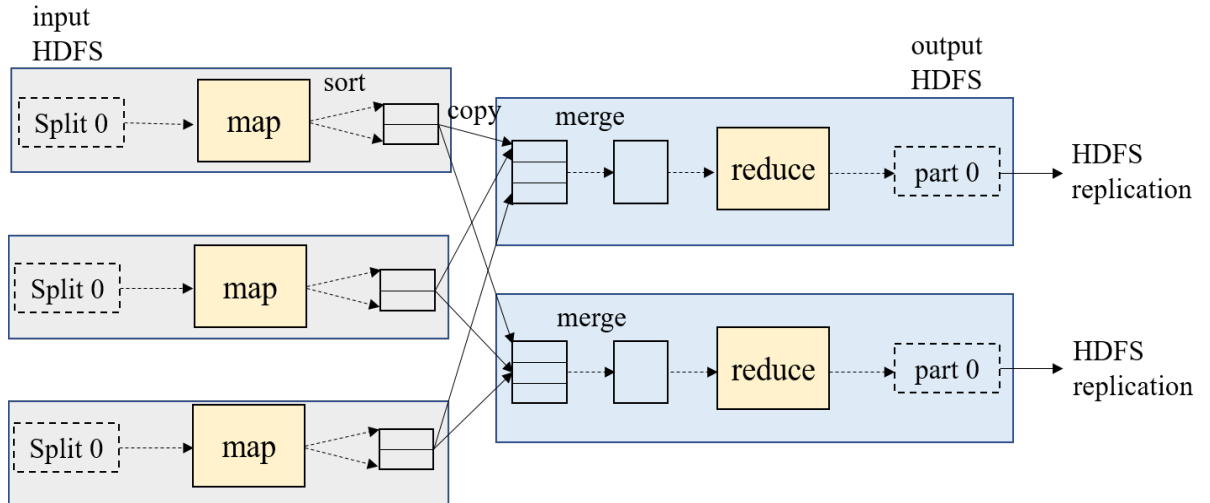


Figure 3.1: Data Flow in MapReduce Process

The Hadoop Distributed File System (HDFS) is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high-throughput access to application data and is suitable for applications that have large datasets. Users or applications can create directories and store files inside those directories. File system namespace allows create, remove, move, and rename access. Further, NameNode maintains file system namespace, and changes to file system namespace or its properties are recorded by NameNode. It also facilitates fault tolerance via a replication mechanism with a customizable replication factor that can be adjusted according to the requirements of the system. Moreover, it provides elastic data storage capability for increasing the amount of data in a cluster.

3.2.2 Hadoop Configuration Parameters

Hadoop 2.7.1 was used in our Hadoop cluster implementation and hundreds of parameters are resided in several configuration files such as `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml` and `yarn-site.xml`. `core-site.xml` is responsible for informing Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop core such as I/O settings that are common to HDFS and MapReduce. `hdfs-site.xml` contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode and the DataNodes. `mapred-site.xml` contains the configuration settings for MapReduce daemons; the job tracker and the task trackers whereas `yarn-site.xml` concerns with the parameters to tune the resource manager and node manager. Configuration parameter settings have significant influence on Hadoop cluster performance in terms of CPU, memory, disk, I/O, and network and parameter settings usually differ greatly on the type of application. Based on previous research' recommendations, we selected some sensitive parameters and optimize those sensitive parameters that can improve the performance of Hadoop cluster in executing Big Data applications. Twenty-four parameters (excluding YARN parameters) were optimized for single objective Hadoop parameter optimization whereas 30 parameters were optimized for bi-objective optimization of Hadoop parameters.

3.2.3 Genetic Algorithms (GA)

GAs are search algorithms based on the mechanics of natural selection and natural genetics[13]. GAs are based on a set of candidate solutions that represent a solution to the desired optimization problem. A solution is a potential candidate for an optimum of the optimization problem. The representation of candidate solutions determines the choice of the genetic operators. Three genetic operators (reproduction, crossover and mutation) are used during optimization through GAs. The basic procedure of the genetic algorithm in [12] is described as follows:

Algorithm: Basic Genetic Algorithm

- Step 1: initialize population
 - Step 2: repeat
 - Step 3: repeat
 - Step 4: crossover
 - Step 5: mutation
 - Step 6: phenotype mapping
 - Step 7: fitness computation
 - Step 8: until population complete
 - Step 9: selection of parental population
 - Step 10: until termination condition
-

3.2.4 Generational and Steady-State Genetic Algorithm

GAs can be classified into two types: generational and steady-state. The generational GA creates new offspring from the members of an old population using the genetic operators and update the population using old population and new offspring population for the next generation whereas steady-state GA update the old population with only one or two offspring at each generation. A replacement/deletion strategy will be replacing the new with the oldest member of the population or the worst member of the population [2]. We used steady state approach in our Hadoop parameter optimization because the proposed system experimentations were conducted on real Hadoop cluster; Hadoop cluster needed to restart on every single parameter changes in the cluster.

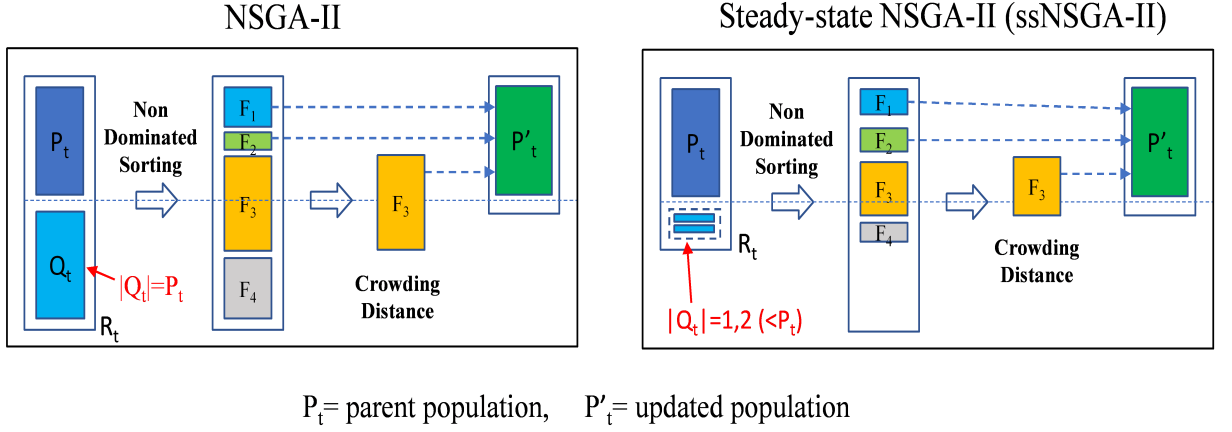


Figure 3.2: Difference between Generational NSGA-II and ssNSGA-II

3.2.5 Steady-State Non-dominated Sorting Genetic Algorithm II

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) [7] is a multi-objective optimization algorithm that has been used for a wide variety of applications in a number of different fields. It uses a fast sorting procedure, an elitism preserving approach, and a parameter-less niching operator. However, according to Nebro et al. [1], steady-state NSGA-II (ssNSGA-II) outperforms this generational NSGA-II on most bi-objective problems in terms of quality assessment, convergence speed, and computing time. Consequently, the proposed system uses steady-state NSGA-II to exploit these benefits during the optimization process. Figure. 3.2 depicts the difference of generational NSGA-II and ssNSGA-II.

3.3 Proposed Optimization System

Search based multi-objective optimization of MapReduce tasks using ssNSGA-II is applied in our proposed system to overcome the Hadoop configuration optimization challenges outlined in Section 3.1. System information details are given in the ensuing sections.

3.3.1 Objective Function

The proposed Hadoop parameter optimization was conducted starting from single-objective optimization to bi-objective optimization.

3.3.1.1 Single-objective optimization

$$\min E_T(p_1, p_2, p_3, \dots, p_j)$$

where,

E_T = execution time of each MapReduce task

p_j = configuration parameter list to control each MapReduce task ($1 \leq j \leq m$)

3.3.1.2 Bi-objective parameter optimization

The objective functions utilized in the proposed system are geared towards minimizing the total execution time to complete the MapReduce tasks and their resource usage, as expressed below:

$$\min t(p), \min c(p)$$

where,

$p = [p_1, p_2, \dots, p_m]$ is the configuration parameter list, instance type

$t(p)$ = execution time of MapReduce job

$c(p)$ = machine instance usage cost

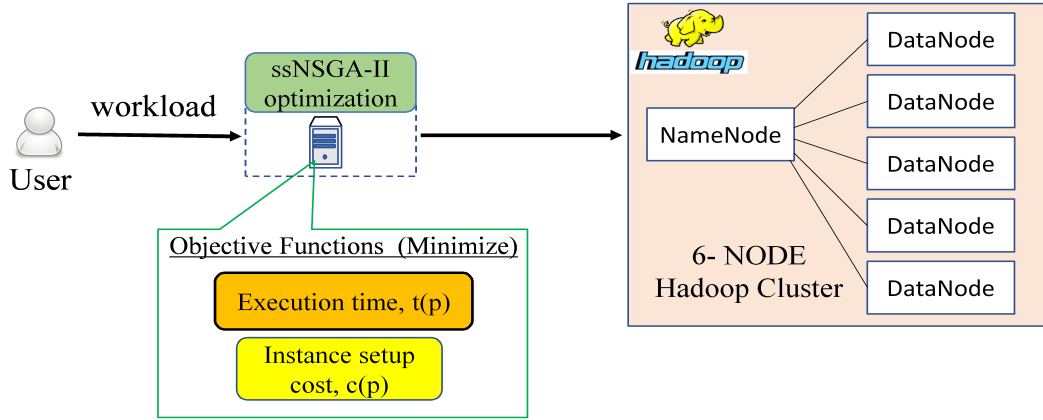


Figure 3.3: Multi-Objective Hadoop configuration optimization system.

3.3.2 System Architecture

The architecture of the multi-objective Hadoop configuration system is shown in Figure. 3.3. Although experiments were done for both single objective and bi-objective optimization, only bi-objective Hadoop parameter optimization will be emphasized in detail system implementation. The operation of the system is as follows. First, workload input is given to the system for parameter optimization, and the system performs all processes in the background, such as simultaneously communicating with Hadoop for fitness calculation and resource usage calculation based on the machine instance type throughout the search space.

3.3.3 Parameter Selection

This section describes the search space (the parameter list) of the proposed optimization system. We categorized the parameters into three groups: HDFS and MapReduce parameters, YARN parameters, and YARN-related MapReduce parameters. The first group of parameters and instance types were then encoded as binary strings during the optimization process. The parameters from the next two groups were automatically assigned their respective values according to the resource available in the machine instance type. A total of 30 configuration parameters from `hdfs-site.xml`, `mapred-site.xml` and `yarn-site.xml` are

Table 3.1: HDFS and MapReduce parameters

No	Parameter Name	Value Range
1	dfs.namenode.handler.count	10, 20
2	dfs.datanode.handler.count	10, 20
3	dfs.blocksize	134217728, 268435456
4	dfs.datanode.max.transfer.threads	4096, 6144, 5120, 7168
5	dfs.datanode.balance.bandwidthPerSec	1048576, 2097152, 4194304, 8388608
6	mapreduce.task.io.sort.factor	10, 20, 30, 40
7	mapreduce.task.io.sort.mb	100, 200, 300, 400
8	mapreduce.map.output.compress	false, true
9	mapreduce.job.jvm.numtasks	1, -1
10	mapreduce.tasktracker.http.threads	40, 45, 50, 60
11	mapreduce.reduce.shuffle.parallelcopies	5, 10, 15, 20
12	mapreduce.map.sort.spill.percent	0.8, 0.9
13	mapreduce.reduce.shuffle.input.buffer.percent	0.7, 0.8
14	mapreduce.reduce.merge.inmem.threshold	1000, 1500, 2000, 2500
15	mapreduce.reduce.shuffle.memory.limit.percent	0.25, 0.5
16	mapreduce.reduce.shuffle.merge.percent	0.66, 0.9
17	mapreduce.reduce.input.buffer.percent	0.0, 0.5

considered in our proposed system. We selected some parameters according to [10] and reports of Hadoop suppliers such as Cloudera and Hortonworks. In addition, YARN and YARN related parameters are quite complex, as they should be in available resource value range of each machine instance type. Tables 3.1 - 3.5 show the selected parameters for the optimization system with their respective value ranges.

We considered four types of machine instances and calculated the cost according to Hokkaido University Academic Cloud instance pricing [15]. Table 3.6 shows the instance pricing used in our resource usage calculation.

3.3.4 System Process Overview

First, random initial chromosome parameter lists are generated to initiate the process. The parameter lists not only include the Hadoop configuration parameters but also the virtual machine instance types for multi-objective optimization purposes. Next, the cost functions are evaluated. Then, the evolution process proceeds until the specified termina-

Table 3.2: YARN and YARN related MapReduce parameters for x-large instance

No	Parameter Name	Value for x-large instance
1	yarn.nodemanager.resource.memory.mb	102400
2	yarn.nodemanager.resource.cpu-vcores	39
3	yarn.scheduler.maximum-allocation-mb	102400
4	yarn.scheduler.minimum-allocation-mb	5120
5	yarn.scheduler.maximum-allocation-vcores	39
6	yarn.scheduler.minimum-allocation-vcores	10
7	mapreduce.map.memory.mb	5120
8	mapreduce.reduce.memory.mb	10240
9	mapreduce.map.cpu.vcores	10
10	mapreduce.reduce.cpu.vcores	10
11	mapreduce.child.java.opts	-Xmx8192m
12	yarn.app.mapreduce.am.resource.mb	10240
13	yarn.app.mapreduce.am.command.-opts	-Xmx8192m

Table 3.3: YARN and YARN related MapReduce parameters for large instance

No	Parameter Name	Value for large instance
1	yarn.nodemanager.resource.memory.mb	26624
2	yarn.nodemanager.resource.cpu-vcores	9
3	yarn.scheduler.maximum-allocation-mb	26624
4	yarn.scheduler.minimum-allocation-mb	2048
5	yarn.scheduler.maximum-allocation-vcores	9
6	yarn.scheduler.minimum-allocation-vcores	3
7	mapreduce.map.memory.mb	2048
8	mapreduce.reduce.memory.mb	4096
9	mapreduce.map.cpu.vcores	3
10	mapreduce.reduce.cpu.vcores	3
11	mapreduce.child.java.opts	-Xmx3276m
12	yarn.app.mapreduce.am.resource.mb	4096
13	yarn.app.mapreduce.am.command.-opts	-Xmx3276m

Table 3.4: YARN and YARN related MapReduce parameters for medium instance

No	Parameter Name	Value for medium instance
1	yarn.nodemanager.resource.memory.mb	10240
2	yarn.nodemanager.resource.cpu-vcores	3
3	yarn.scheduler.maximum-allocation-mb	10240
4	yarn.scheduler.minimum-allocation-mb	2048
5	yarn.scheduler.maximum-allocation-vcores	3
6	yarn.scheduler.minimum-allocation-vcores	1
7	mapreduce.map.memory.mb	2048
8	mapreduce.reduce.memory.mb	2048
9	mapreduce.map.cpu.vcores	1
10	mapreduce.reduce.cpu.vcores	1
11	mapreduce.child.java.opts	-Xmx1638m
12	yarn.app.mapreduce.am.resource.mb	2048
13	yarn.app.mapreduce.am.command.-opts	-Xmx1638m

Table 3.5: YARN and YARN related MapReduce parameters for small instance

No	Parameter Name	Value for small instance
1	yarn.nodemanager.resource.memory.mb	3072
2	yarn.nodemanager.resource.cpu-vcores	1
3	yarn.scheduler.maximum-allocation-mb	3072
4	yarn.scheduler.minimum-allocation-mb	1024
5	yarn.scheduler.maximum-allocation-vcores	1
6	yarn.scheduler.minimum-allocation-vcores	1
7	mapreduce.map.memory.mb	1024
8	mapreduce.reduce.memory.mb	1024
9	mapreduce.map.cpu.vcores	1
10	mapreduce.reduce.cpu.vcores	1
11	mapreduce.child.java.opts	-Xmx819m
12	yarn.app.mapreduce.am.resource.mb	1024
13	yarn.app.mapreduce.am.command.-opts	-Xmx819m

Table 3.6: Machine Instance Pricing

No	Instance Type	CPU (No. of cores)	Memory (GB)	Price per second (yen)
1	Extra large	40	128	0.0160
2	Large	10	30	0.0039
3	Medium	4	12	0.0016
4	Small	1	3	0.0004

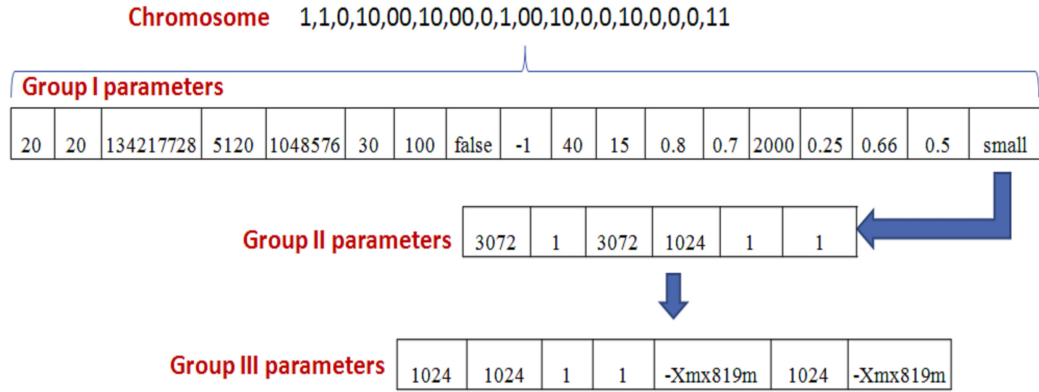


Figure 3.4: Conversion from Binary Chromosome to Real Parameters.

tion criterion is met. At this point, the Pareto front solutions from the evolution process are retrieved and the optimized configuration settings that give the optimal job execution time and optimal type of instance for the cluster at the Pareto front is recommended for use in future processing of similar workloads in the cluster.

3.3.5 Chromosome Representation

The system uses a binary encoding scheme, the most common representation of chromosomes in GA. In this system, the genes in the chromosomes are in the form 0, 1 with each gene—a single bit or two consecutive bits—representing its respective Hadoop configuration parameter values. Two bits are used for parameters with four combination values and one bit for two combination parameters. Further, strings of 18 parameter values (genes) are encoded as 26-bit binary chromosome strings (24 bits for Hadoop configuration parameters plus two bits for instance type for the cluster) that are then used in the parameter optimization process of the proposed system. The chromosome decoding process is illustrated in Figure. 3.4.

3.3.6 Fitness Functions

In this system, execution time cost and instance resource usage cost are considered as the fitness values for multi-objective optimization. However, dynamic increases and decreases in the number of machine instances are not considered. Execution time cost is the total elapsed time of the MapReduce workload. This is given as follows:

$$t(p) = t_{wc} \quad (3.1)$$

where,

t_{wc} =actual workload execution time

The machine instance cost is calculated as follows:

$$c(p) = (SP * NS) * t(p) \quad (3.2)$$

where,

SP = instance price

NS = number of machine instances

3.4 Multi-Objective Hadoop Parameter Optimization Algorithm

This section presents the single objective optimization through steady-state genetic algorithm and ssNSGA-II based optimization algorithms for bi-objective optimization, which are the major components of the proposed parameter optimization approaches.

Algorithm 3.1: Steady-State GA based Hadoop Configuration Optimization
(single-objective)

Input: Population size, N; number of generations, G.

Output: List of optimized configuration solutions, Copt C_{opt} .

Step 1: Generate random initial population, IP.

1.1 Generate binary chromosome randomly $C_i (i = 1, 2, \dots, N)$.

1.2 Calculate the fitness of each random chromosome $f_i = E_T(Ci)$.

Step 2: Evolve population (P, G).

2.1 Select parents at random, (P_1, P_2) .

2.2 Perform two-point dynamic crossover operation on parents (P_1, P_2) with probability $P_c = 1$ to produce offsprings C_1 and C_2 .

2.3 Perform mutation operation with probability $P_m = 0.01$ on random points applied to the offsprings C_1 and C_2 .

2.4 Calculate fitness of new offspring C_1 and C_2 and replace the best two of (P_1, P_2, C_1, C_2) with the worst two chromosomes in the population P

Step 3: Repeat Step 2 until the predefined number of generations is reached

Step 4: Output the list of optimal Pareto parameter solutions, C_{opt} .

Algorithm 3.2: ssNSGA-II based Hadoop Configuration Optimization (bi-objective)

Input: Population size, N; number of generations, G; number of objectives,

O; mutation probability, MP; crossover probability, CP.

Output: List of optimized configuration solutions, Copt C_{opt} .

Processing

Step 1: Generate random initial population, IP.

1.1 Randomly generate integer chromosome $C_i (i = 1, 2, \dots, N)$.

1.2 Calculate the fitness of each random chromosome

$$f_i = t_{wc} = E_T(C_i) \text{ and } \text{instancecluster cost}(p).$$

1.3 Rank population P according to non-domination

Step 2: Evolve population (N, P, G).

2.1 Select parents using tournament selection, (P_1, P_2) .

2.2 Perform two-point dynamic crossover operation on parents (P_1, P_2) with probability P_c to produce offspring population C_1 and C_2 .

2.3 Perform mutation operation with probability MP on random points applied to the offsprings

2.4 Calculate fitness of new offspring population and update population P

2.5 Use non-dominated sorting to divide $P \leftarrow P \cup \{x^C\}$ into several non-domination levels $F_1, F_2 \dots, F_l$.

2.6 Calculate Crowding Distance of all solutions

2.7 Identify the worst solution $x' \in F_l$ and set $P \leftarrow P' \setminus \{x'\}$.

Step 3: Repeat Step 2 until termination condition is met.

Step 4: Output the list of optimal solutions at Pareto front, C_{opt} .

3.5 Performance Evaluation

This section outlines the experimental environment and presents the experimental results obtained for various workloads. Both cluster testing environments were configured with updated Hadoop 2.7.1 and JDK 1.8.0 and Python programming language was used for the implementation.

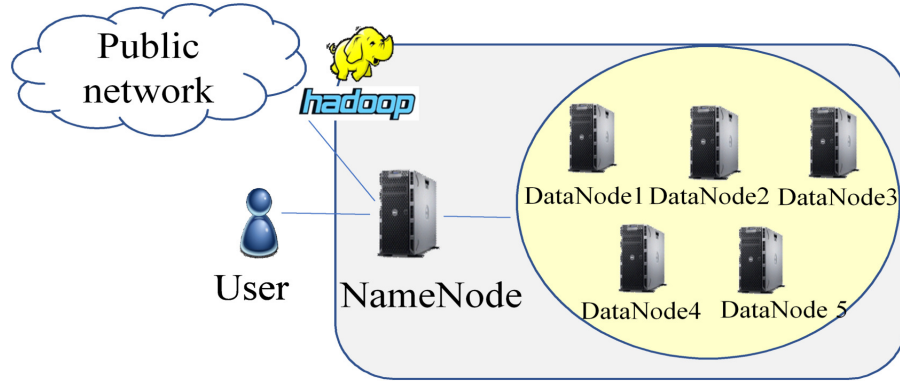


Figure 3.5: Hadoop Cluster Experimental Environment

3.5.1 Hadoop Cluster Configuration

Two Hadoop cluster testing environments were used for single-objective and bi-objective optimization of Hadoop parameters. Single-objective Hadoop configuration optimization testing was conducted using 24 VMs on Hokkaido University academic cloud system. The system comprises one NameNode, the front end for the cluster system, and 23 Datanodes. Each node comprises 4GB RAM , and Intel Xeon E7 CPU, and 100 GB of storage.

For bi-objective optimization experiments, a Hadoop cluster comprising six machines was implemented. The cluster included one NameNode, which served as the front end for the cluster, and five DataNodes. Each node had 40 cores CPU, 128 GB RAM and 4 TB of storage. Figure 3.5 depicts the cluster architecture used in the experiments.

3.5.2 Experimental Benchmarks

In this section, we describe the benchmark suite that we used for the evaluation of our proposed system. We chose the HiBench 4.0 benchmark [9], developed by Intel, as it is a new, realistic, and comprehensive benchmark suite that properly evaluates and characterizes the Hadoop framework. Table 3.7 shows the benchmark programs used in the experiments, along with their respective input data size.

Table 3.7: Benchmark Program used in the Experiment

Type	Workload	Input Size
MicroBenchmark	Sort	2.98GB
	TeraSort	
	Wordcount	
Web Search	Pagerank	5000 pages

3.5.3 Experimental Analysis

This section describes the experimental results obtained from single-objective and bi-objective Hadoop parameter optimization when running benchmark applications such as, sort, terasort, wordcount, and pagerank.

3.5.3.1 Experimental results for single-objective optimization

Table 3.8 shows the result comparison of proposed optimized parameter setting using steady state GA approach with the default parameter settings. The experiments were conducted with the population size 30, mutation probability 0.2 and crossover probability 1 using steady-state genetic algorithm approach. High crossover probability was used during experiments to support higher possibility of Hadoop parameter variations. According to the experimental results obtained after 5 runs, the proposed system could identify the optimized parameter settings that satisfy the objective function of reducing the execution time of workload applications. The experimental results indicate that the proposed approach could identify the optimal Hadoop parameter settings that speed up execution time. For sort workload, the execution time can be reduced up to 2.29 times compared with the default Hadoop parameter setting. The system effectiveness can also be seen from the experimental results for other workload applications such as terasort with 1.16 times improvement, wordcount with 2.21 times improvement, pagerank with 1.25 improvement and kmeans with 1.61 improvment. No YARN paramters were considered in this single-objective experimentation and these parameters considerations were included in the next bi-objective Hadoop optimization system.

Table 3.8: Single-Objective Hadoop Parameter Optimization Results

Workload Type	Workload Name	Execution time (Default Setting)	Execution time (Optimized Setting)
MicroBenchmark	Sort	358.405 sec	156.134 sec
	TeraSort	114.986 sec	98.581 sec
	Wordcount	596.963 sec	269.602 sec
Web Search	Pagerank	475.212 sec	379.867 sec
Machine Learning	Kmeans	777.584 sec	483.037 sec

3.5.3.2 Experimental results for bi-objective optimization

Bi-objective Hadoop parameter optimization was the extended version of single objective Hadoop parameter optimization and experiments were conducted on high specification Hadoop cluster. Figures 3.6 - 3.9 shows the optimal Hadoop parameter settings obtained using the proposed optimization approach. The experiments were conducted with a population size of 30, mutation probability 0.1, crossover probability 0.9 for 50 generations. Three groups of Hadoop parameters (HDFS and MapReduce parameters, YARN parameters and YARN related MapReduce parameters) were considered in this experimentations. The results indicate that sort, terasort and wordcount workloads depend much on HDFS and MapReduce parameters because the Pareto optimal solutions obtained after 50 generations give sparse solutions even for each machine instance type. For pagerank application, the system gives overlap solutions which means changes in HDFS and MapReduce parameters have weak effect on pagerank application compared with YARN and YARN related MapReduce parameter changes.

As the experiments were conducted on real Hadoop cluster, it took long time to identify the optimal Hadoop parameter settings for all workload applications (specifically, about 9 hours for sort, 6 hours for terasort, 9 hours for wordcount and 14.5 hours for pagerank applications).

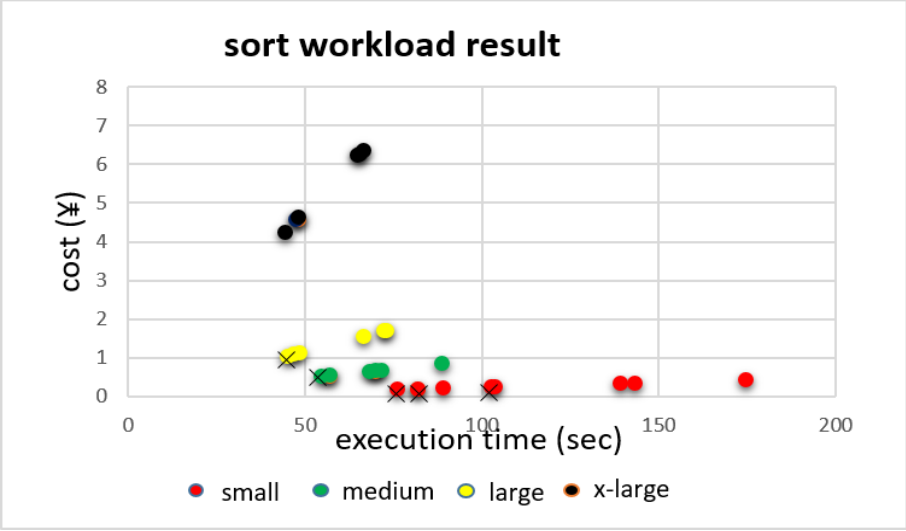


Figure 3.6: Optimization Result for Sort Workload.

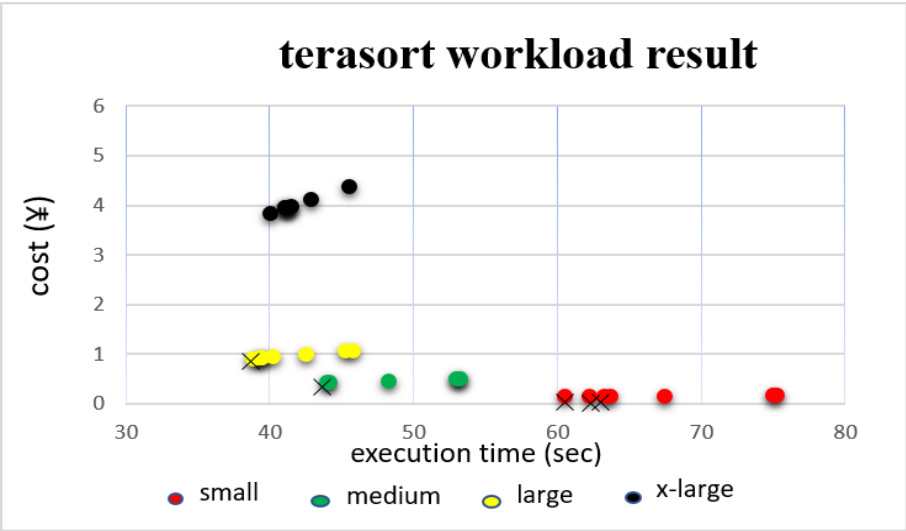


Figure 3.7: Optimization Result for Terasort Workload.

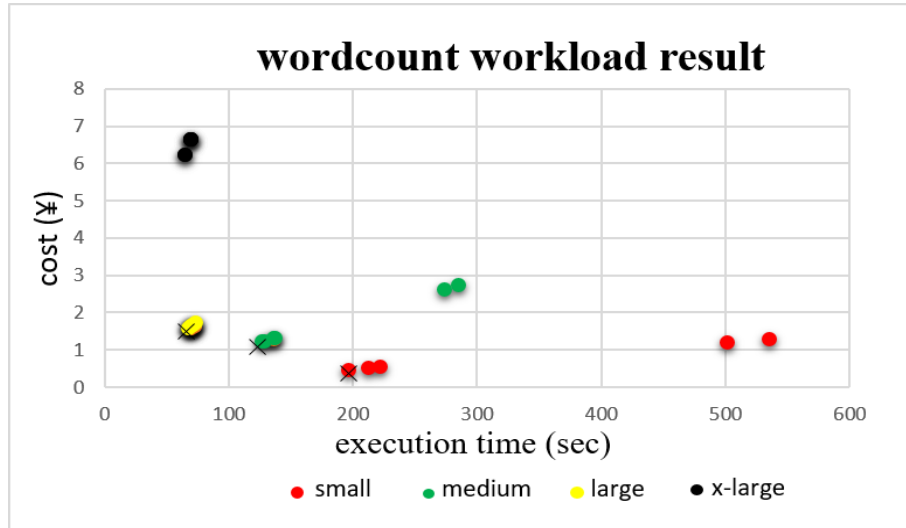


Figure 3.8: Optimization Result for Wordcount Workload.

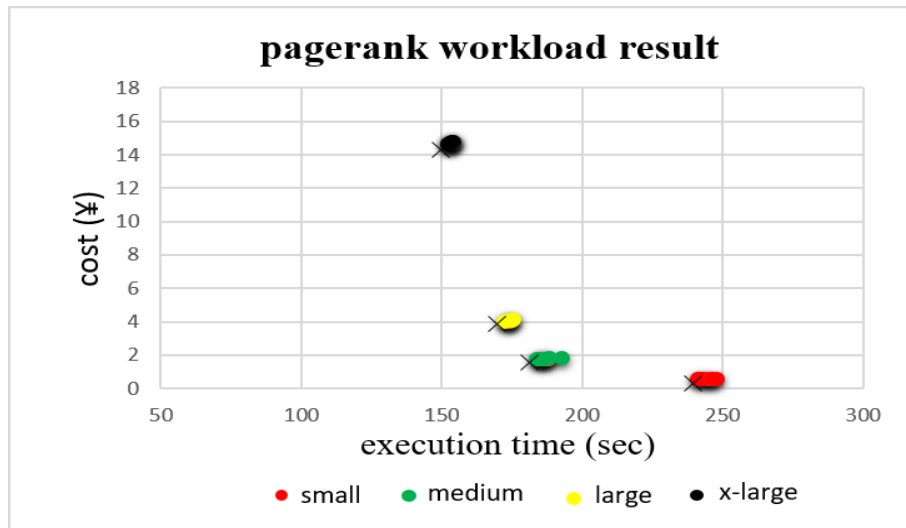


Figure 3.9: Optimization Result for Pagerank Workload.

3.6 Related Work

Hadoop parameter configuration optimization is an interesting research area knowledge harvesting that is currently attracting many scientists and engineers in this Big Data era. According to the literature, there are four major approaches taken to parameter tuning: (1) rule of thumb based tuning, (2) cost based tuning, (3) machine learning based tuning, and (4) search based tuning. In these methods, rule of thumb based tuning is used to provide general tuning settings for various applications, which may result in applications performing as well as they should because applications are diverse in nature. Lin et al. [11] proposed a cost based performance model for Hadoop MapReduce processing in which they identify the I/O cost, network cost, and CPU cost during MapReduce processing using definitions and three kinds of parameters; namely, cluster parameters, MapReduce system parameters, and MapReduce job parameters. Their evaluation results showed that the system is effective; however, they stress that further refinement and consideration are necessary if the combine operation is included in the MapReduce application. The Starfish model [4] uses the average CPU and I/O cost of reads and writes to estimate the average map execution time by profiling jobs with a single configuration. In this case, the model predicts the same map time as the number of reduce tasks from 64 to 256. Gunther [3] is a GA search based parameter tuning approach that aggressively identifies parameter settings that result in near-optimal job execution times. The results of evaluation experiments conducted indicate that this approach can obtain near-optimal performance within 30 trials using six configuration parameters with specified value range. MRONLINE [8] is a proposed online parameter tuning method that utilizes a gray box-based smart hill climbing algorithm that can efficiently converge to a near-optimal configuration with high probability. This approach was shown to dynamically improve performance up to 30% compared with default YARN. Kim et al. [6] proposed an optimal parameter setting identification method that employs a simple sampling technique. The system can automatically identify

sets of top parameters that reduce the execution time of MR tasks by 32 percent compared to default set of parameters. The above approaches are single objective optimization on execution time of MapReduce tasks. Our proposed approach considers bi-directional optimization on time and machine instance cost of the cluster for MapReduce application processing.

3.7 Conclusion

Hadoop MapReduce parameter configuration has a significant impact on application performance. However, parameter configuration optimization to speed up MapReduce processing in Hadoop is a daunting and time-consuming task because it is very difficult to identify the parameters that significantly affect the performance of a particular application. Further, the types of optimized machine instances that are best for executing MapReduce applications in a cluster are also very difficult to determine. This proposed system performs offline Hadoop configuration optimization starting with single objective optimization (execution time minimization) and extends the model to support two objective functions using an ssNSGA-II based search strategy. The bi-objective parameter optimization experiments were conducted using a six-node real Hadoop cluster and the results indicate that the proposed system improves application performance of big data applications through the proposed Hadoop parameter optimization model.

Bibliography

- [1] A. J. Nebro and J. J. Durillo. *On the Effect of Applying a Steady-State Selection Scheme in the Multi-Objective Genetic Algorithm NSGA-II*. In NAO, vol 193, pp 435–456. Springer, 2009.
- [2] F. Vavak and T. C. Fogarty, *A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments*. AISB International Workshop on Evolutionary Computing, UK, Springer, 1996.
- [3] G. Liao, K. Datta, and T. L. Wilke *Gunther: Search based auto tuning of MapReduce*. Intel Labs, Hillsboro, Oregon, USA, LNCS8097, pp 406-419, Springer-Verlag Berlin Heidelberg, 2013.
- [4] H. Herodotou, H. Lim, G. Luo, N. Borisov, and L. Dog *Starfish: A self tuning system for big data analytics* 5th Biennial Conference on Innovative Data Systems Research (CIDR’11), Asilomar, California, USA, vol. 11, 2011.
- [5] J. Dean and S. Ghemawat *MapReduce: Simplified Data Processing on Large Clusters*. Common. ACM, vol. 51, no. 1, pp. 107-113, 2008.
- [6] J. Kim, T. K. A. Kumar, K. M.. George, and N. Park *Performance evaluation and tuning for MapReduce computing in Hadoop distributed file system* Proc. 13th International Conference on Industrial Informatics (INDIN), IEEE Press, pp. 62-68, July 2015

- [7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan *A fast and elitist multiobjective genetic algorithm: NSGA-II* IEEE Transactions on Evolutionary Computation, vol. 2, Issue 6, pp. 182-197, 2002.
- [8] M. Li, L. Zeng, S. and Meng *MROnLINE: MapReduce online performance tuning*. Proc. ACM HPDC 14, June 23-27 Vancouver, Canada, 2014.
- [9] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang *The HiBench benchmark suite: Characterization of the MapReduce-based data analysis* Proc. 26th International Conference on Data Engineering Workshops (ICDEW), March 2010, pp. 41-51.
- [10] S. Joshi *Hadoop Tuning Guide*. Advanced Micro Devices, Tech.Rep., Oct 2012.
- [11] X. Lin, Z. Meng, C. Xu, and M. Wang *A practical performance model for Hadoop MapReduce*. Proc. International Conference on Cluster Computing Workshops, pp. 231-239, Beijing, Sept 2012.
- [12] O. Kramer *Genetic Algorithm Essentials*. Studies in Computational Intelligence 679, doi 10.1007/978-3-319-52156-5_2, 1007, 2017.
- [13] D. E. Goldberg *GENETIC ALGORITHMS in Search, Optimization & Machine Learning*. Addison-Wesley Publishing Company, Inc, 1989.
- [14] T. White *Hadoop; The Definitive Guide, Fourth Edition*. O'Reilly Media, Inc, 2015.
- [15] Hokkaido University Academic Cloud System
<http://www.hucc.hokudai.ac.jp/hosting-server.html>. Accessed January 2016.

Chapter 4

Multi-Objective Optimization of Scientific Workflows in the Cloud

This chapter presents the details of multi-objective scientific workflow execution optimization on the public cloud system model.

4.1 Introduction

Scientific experiments in fields such as bioinformatics, astronomy, elementary particle physics and life science requires the storing and processing of Big Data. Grid computing systems are widely used in scientific applications (e.g., [3], [27]) as they provide parallelism through high-performance computational infrastructure. However, nowadays, cloud computing has supplanted grid computing for the execution of scientific applications as it provides enormous amounts of scalable computational resources on demand. Further, the adoption of cloud computing to process these scientific data has increased recently as cloud enables data correlations, pattern mining, data predictions and data analytics in a cost-efficient manner.

In cloud computing, resources such as networks, storage, applications, and servers can

be allocated from a shared resource pool with minimal management or interaction [28]. In general, cloud computing provides Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) in a pay-per-use model. Of these services, IaaS is the most frequently utilized as it provides customers with an elastic facility to provision or release virtual machines (VMs) in the cloud infrastructure [30]. The elastic nature of cloud facilitates changing of resource quantities and characteristics to vary at runtime, thus dynamically scaling up when there is a greater need for additional resources and scaling down when the demand is low [5]. Because of its elastic resource provisioning mechanism, the cloud is widely used in several areas including for large-scale scientific applications. With improved service support and cloud bursting technologies, users do not need to provision their resources for the worst-case scenarios. In the context of cloud computing, an application's execution cost means the monetary cost of renting resources from the cloud service provider. Addressing the issues of minimizing the resource usage with the best performance is an important research area in an IaaS cloud.

Scientific applications consist of thousands of tasks requiring heavy computation and data transfer. Workflow tasks also have certain dependencies during execution. Modeling of these scientific applications is necessary for effective processing. The most widely used representation model is in the form of Directed Acyclic Graph (DAG) in which the structure of the workflow indicates the order of execution of tasks. Our optimization model deals with scientific workflow in the form of DAG during optimization experimentations. In addition, it is assumed that the monetary cost for workflow execution in the cloud is based on the amount of resources used, i.e., task execution cost correlated to the total number of CPU cycles for all tasks. By using cloud infrastructure, customers can also reduce the processing cost of workflow applications with the aid of the cloud's pay-per-use model. Moreover, a task that shares the same time interval with a previous task hosted in the same instance might not produce extra cost, reducing the overall workflow execution cost.

Even in the case of processing workflow applications in the cloud, workflow scheduling

is very important because it can significantly affect workflow makespan. Although much research has been conducted on the workflow scheduling problem to minimize makespan during execution, much of them only focused on single-objective optimization to reduce workflow makespan using an effective scheduling mechanism. According to Zhu et al. [31], even though multi-objective workflow scheduling has been conducted on grid platforms with the aid of several techniques such as NSGA-II [14], SPEA2 [7], and PAES [12], these techniques cannot be applied directly to the cloud. Consequently, they proposed multi-objective optimization of workflow scheduling in the cloud within specified deadline and budget constraints. However, in their approach, they assumed that the number of machine instances is equal to the number of tasks in the workflow and that all tasks are parallelizable. In this case, their approach does not emphasize the dependency among workflow tasks during execution. The approach conducted workflow optimization via makespan minimization with a fixed number of VMs and no elasticity during workflow execution. Meena et al. [13] propose a meta-heuristic cost effective genetic algorithm that integrates a novel scheme for encoding, initial population and genetic operators in the original genetic algorithm (GA) approach to minimize the execution cost of the workflow while meeting the deadline in cloud computing environment. Cui et al. [9] formulated the task scheduling problem in the cloud as a constrained multi-objective optimization problem over reliability, makespan and flowtime using a GA based chaotic ant swarm algorithm. Markov-based reliability analysis of the cloud services is conducted before resource allocation to the virtual machine. But the approach lacks resource sharing within a single virtual machine by assuming that each node can process only one subtask at a time.

Khajemohammadi et al. [11] proposed a leveled fast workflow scheduling strategy, that minimizes cost and time in a Grid environment. However, in that approach, resource utilization is only possible within predefined set of service resources, which is irrelevant because of the dynamicity and elasticity of the cloud. Moreover, a service only processes one task at a time within a level. Our approach differs from the previous work in several

aspects 1) instead of a predefined set of service resources, randomized search on the number of VMs and various machine instance type is conducted during execution to benefit from cloud elasticity and monetary facility, 2) more than one task can share a single virtual machine instance for better resource utilization, 3) balanced task assignment is conducted to balance the load in each machine instance as well as to reduce the wait time to proceed to the next level of tasks; 4) minimizing failure of machine instances to improve the reliability during workflow execution is considered. In that way, the system searches for the Pareto-optimal solutions that exhibit the best performance with minimized failure at the lowest financial cost.

Our optimization idea is based on the widely used scientific workflows from the Pegasus project [34], which have control and dependencies. The continuous experiment starts with developing unconstrained bi-objective optimization model followed by unconstrained three-objective objective optimization model and constrained four-objective optimization model. The experimental results obtained verify the efficacy of our model effectiveness. The system implementation now conducts optimization on three minimization objectives (makespan, cost, failures) and one maximization objective with 25 virtual machine instance types public cloud system model. Third, we perform experiments and analysis on three real-world workflows using SPEA2, NSGA-II and NSGA-III algorithms to show the system effectiveness on current resource optimization system for Big Data applications.

The remainder of this chapter is organized as follows. Section 4.2 presents the theoretical background associated with scientific workflows, SPEA2, NSGA-II and NSGA-III. Section 4.3 outlines the proposed system, and the objective functions used. Section 4.4 presents the proposed workflow execution optimization system. Section 4.5 describes the resource optimization algorithms used: SPEA2, NSGA-II, and NSGA-III. Section 4.6 discusses the performance evaluation conducted. Section 4.7 presents related work. Finally, Section 4.8 concludes the proposed resource optimization approach.

4.2 Background

This section presents the relevant theoretical background associated with the research. First, scientific workflows and their features are discussed. Then, the multi-objective evolutionary algorithms SPEA2 [7], NSGA-II [14] and NSGA-III [10] are presented.

4.2.1 Scientific Workflows

Scientific workflows have emerged as an alternative to script programming for performing in-silico experiments [25]. They are essential to facilitate and automate the processing of high-volume scientific data in large distributed computing structure such as Grid [32]. These workflows describe the set of tasks needed to carry out computational experiments and provide scientists with the ability to expose, share, and reuse their work. The goals of these workflows [1] are

- To save human cycles by enabling scientists to focus on domain-specific aspects of their work, rather than dealing with complex data management and software issues;
- To save machine cycles by optimizing workflow execution on available resources.

Scientific workflows are created with a corresponding workflow editor that usually provides a repository with predefined workflow activities. In general, a scientific workflow is modeled as a directed acyclic graph (DAG). A DAG workflow, W is defined as $W = (T, D)$, where $T = \{T_0, T_1, \dots, T_n\}$ is a set of tasks and $D = \{(T_i, T_j) | T_i, T_j \in T\}$ is a set of data or control dependencies [7]. Each task has its respective execution time, task name, input data size, and resulting output data size.

For a DAG with multiple entries and exits, it is necessary to add a pseudo T_{entry} and or a pseudo T_{exit} with no control and dependencies. In this study, we assumed that all DAG workflows have only one T_{entry} and one T_{exit} , and optimization is considered for the actual processing steps inside the workflow. A DAG is represented in either XML or JSON format

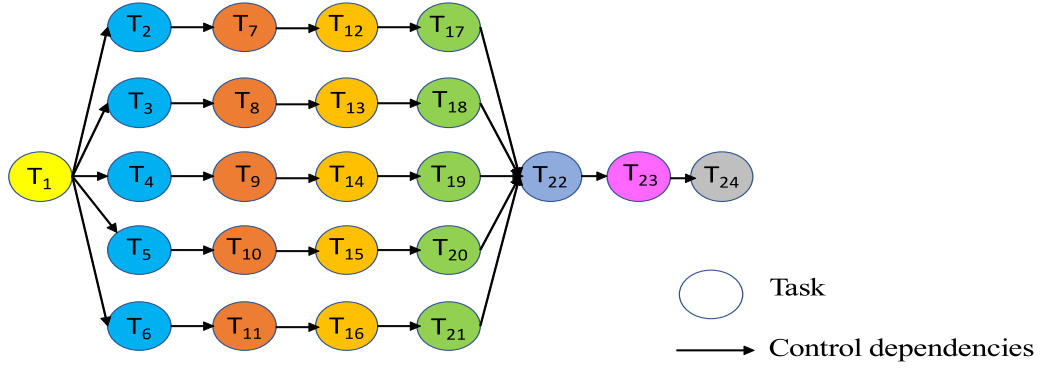


Figure 4.1: An Example DAG Workflow.

and parsing is required to get the necessary information for further processing. A simple DAG is illustrated in Figure 4.1 with respective task labels.

4.2.2 Constrained Optimization

Optimization is a fascinating area of study not only because of its algorithmic and theoretical content, but also because of its universal applicability. Evolutionary algorithms are used to find the optimal solution for a difficult problem with several feasible solutions. EAs can make a task manageable and at least find a good solution. Thus, EAs are widely used in solving complex real-world problems. Moreover, real-world optimization problems almost always have constraints. The optimizing value of the independent variable almost always occurs on the constraint boundary in real world optimization problems. Constraints are therefore important in almost all real-world optimization problems. Not only are real-world optimization problems constrained but they are also multi-objectives [31]. It is obvious that people normally expect the best solution from every perspective as possible. We are interested in minimizing or maximizing more than one measure simultaneously. So, constrained multi-objective optimization through EAs help us to solve complex multi-objective real-world problems.

4.2.3 Multi-Objective Evolutionary Algorithms (MOEAs)

MOEAs have been proposed for finding multiple Pareto-optimal solutions in one single simulation run. MOEAs are an effective way to find the Pareto-optimal solutions among conflicting objectives. They produce non-dominated solutions with respect to minimization or maximization of objectives for user satisfaction. There exist several MOEAs in the literature. The popular algorithms: SPEA-2, non-dominated sorting genetic algorithm II (NSGA-II), and non-dominated sorting genetic algorithm III (NSGA-III) were used during this study. A brief explanation of each of these algorithms is given in the ensuing section.

4.2.3.1 Improved Strength Pareto Evolutionary Algorithm 2 (SPEA2)

SPEA2 [7] is an improved version of the original Strength Pareto Evolutionary Algorithm (SPEA) by means of an improved fitness assignment scheme, nearest neighbor density estimation technique, and new truncation method. The algorithm starts with a population and an archive (external set). Then it performs iterations for the evolution process. During the update operation in the evolution process, filling up with the dominated individuals or truncating individuals if necessary to fit in the fixed-size archive occurs. Unlike SPEA, SPEA2 only considers members of the archive to participate in the mating selection process. The objective of the algorithm is to locate and maintain a front of non-dominated solutions, ideally a set of Pareto optimal solutions. This is achieved by using an evolutionary process (with surrogate procedures for genetic recombination and mutation) to explore the search space, and a selection process that uses a combination of the degree to which a candidate solution is dominated (strength) and an estimation of the density of the Pareto front as an assigned fitness. An archive of the non-dominated set is maintained separate from the population of candidate solutions used in the evolutionary process, providing a form of elitism.

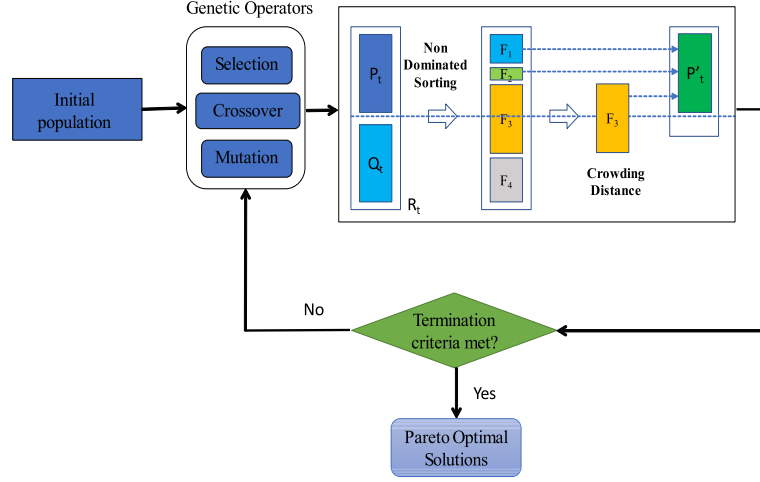


Figure 4.2: Process Flow of NSGA-II Optimization.

4.2.3.2 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

In 1995, Non-dominated sorting Genetic Algorithm, (NSGA) [20], which improved computational complexity via a non-elitism approach and a sharing parameter, was developed. In 2002, Deb et al. developed NSGA-II [14], an improved version of NSGA rectifies the problems found in NSGA. NSGA-II is a popular multi-objective optimization algorithm that is widely used in several application domains. The main features of NSGA-II are low computational complexity, parameter-less diversity preservation and an elitism approach. The major components of NSGA-II include fast non-dominated sorting and crowding distance assignment.

Fast Non-dominated Sorting

Elitism is introduced in NSGA-II by storing all non-dominated solutions discovered, starting with the initial population [2]. Elitism enhances the convergence properties towards the Pareto-optimal set. In order to sort the individuals in the population according to the level of non-domination, each solution must be compared with every other solution to determine if it is dominated. Firstly, two entities are calculated for each solution: 1) n_p , the number of solutions that dominate solution p and 2) S_p , a set of solutions that solution p dominates. All solutions that have $n_p = 0$, i.e., non-dominated solutions are identified and

placed in a current front list as first non-dominated front list. It is also regarded as the first non-dominated front. Again, for each solution in the current front, each member in its S_p is visited and n_p is reduced by one. At this point, if any new member of $n_p = 0$ is identified, that member is placed in a separate list Q, the second non-dominated front. The above procedure is continued with list Q to identify the third non-dominated front. The process continues until all fronts are identified. Fast non-dominated sorting on a population returns a list of non-dominated fronts F_r and the population is updated starting with the members in the best non-dominated front followed by the second-best non-dominated front, and so on until the specified population size is reached. Moreover, the fast-nominated approach reduces the complexity of the naive approach from $O(MN^3)$ to $O(MN^2)$. Figure.4.3 illustrates the process of Non dominated sorting in NSGA-II algorithm.

Crowding Distance Algorithm

After completing non-dominated sorting, crowding distance is assigned. In this process, all the individuals in the population are assigned a crowding distance value and individual selection is conducted based on rank and crowding distance. In other words, crowding distance involves finding the Euclidean distance between each individual in a front based on their m objectives in an m -dimensional hyperspace. The individuals in the boundary are always selected because they have infinite distance assignment [26]. Figure 4.4 shows how crowding distance operation is conducted to rank the several front solutions during optimization. Crowding distance comparison compares the individuals within the same front and distance calculation is conducted as follows:

For each front F_i , n is the number of individuals:

1. Initialize the distance $F_i(d_j) = 0, j = 1, \dots, n$

2. for each objective function m

$$I = \text{sort}(F_i, m) \quad \# \text{Sort the individuals on objective } m$$

$$I(d_1) = I(d_n) = \infty \quad \# \text{assign infinite distance to boundary values}$$

for $k = 2$ to $(n - 1)$

$$I(d_k) = I(d_k) + \frac{I(k+1) \cdot m - I(k-1) \cdot m}{f_m^{max} - f_m^{min}}$$

$I(k).m$ is the m^{th} objective value of the k^{th} individual in I

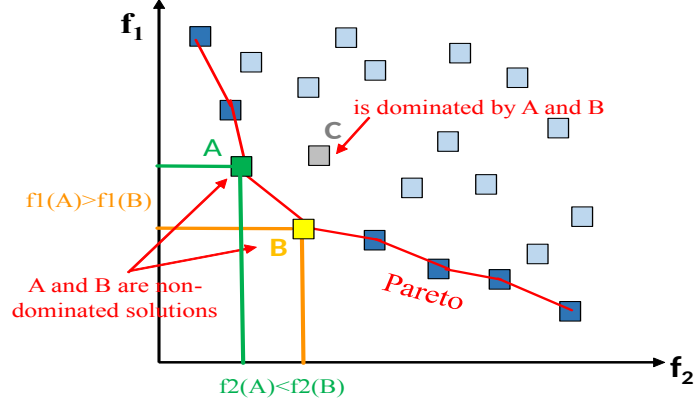


Figure 4.3: Non-dominated Sorting in NSGA-II.

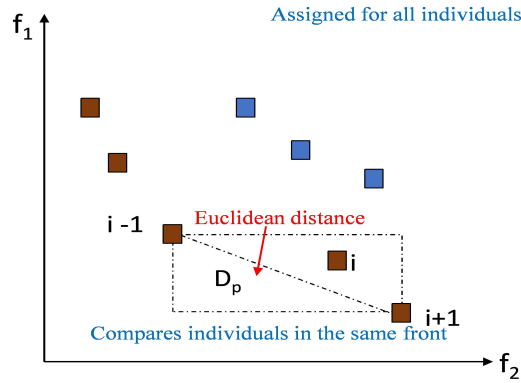


Figure 4.4: Crowding Distance Operation.

The solutions found in NSGA-II are said to be Pareto optimal if they are not dominated by any other solutions in the solution space. Further, the set of all feasible non-dominated solutions in the solution space is referred to as the Pareto-optimal set, and for a given Pareto-optimal set, the corresponding objective function values in the objective space are called the Pareto front. The ultimate goal of multi-objective optimization is to identify the solutions in the Pareto-optimal set. In this proposed optimization system, we use NSGA-II as it is one of the most popular optimization approaches in several domains. Figure 4.2

depicts the process flow of NSGA-II optimization.

4.2.3.3 Non-dominated Sorting Genetic Algorithm III (NSGA-III)

The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) is widely applied in various practical optimization problems over the years. It works well in optimizing until three objectives but it becomes difficult to handle many objectives optimization problems involving four or more objectives. To solve this problem, Jain and Deb proposed a new extension of NSGA-II (MO-NSGA-II) in 2012. Later, they extended the MO-NSGA-II to NSGA-III [10] to solve many objectives optimization as constrained optimization problems.

Like in NSGA-II, NSGA-III starts with initial population P_t and a set of reference points Z^r . Then the binary tournament selection, crossover and mutation operators are applied to create an offspring population Q_t . Then P_t and Q_t are combined to be R_t and sorted according to their domination level and the best N members are selected to be used for the next generation. The difference in NSGA-III is that it starts with a set of reference point Z^r . After non-dominated sorting, all acceptable front members and the last front F_l which could not be completely accepted are saved in a set S_t . Members in S_t/F_l are selected right away for the next generation. The remaining members are selected from F_l such that a desired diversity is maintained in the population. Instead of using crowding distance measure for selecting well-distributed set of points, NSGA-III used reference points to select the remaining members. Figure 4.5 illustrates the reference point calculation process. During that process, objective values and reference points are first normalized so that they have an identical range. Thereafter, orthogonal distance between a member in S_t and each of the reference lines is calculated. The member is then associated with the reference point having the smallest orthogonal distance. Next, the niche count ρ for each reference point, defined as the number of members in S_t/F_l that are associated with the reference point, is computed for further processing. The reference point having the minimum niche

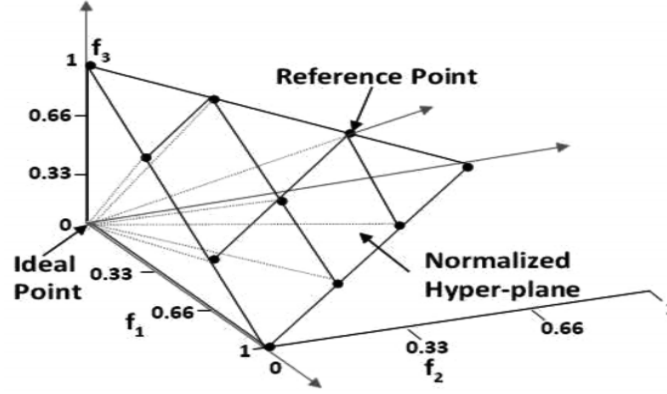


Figure 4.5: Reference Points for Three-Objective Optimization [29].

count is identified and the member in front last front F_l is associated with the identified reference point is increased by one and the procedure is repeated to fill up population P_{t+1} [10]. Then, the evolution process starts again with the updated population and it continues until the defined number of generations is reached.

4.3 Proposed Optimization System Model

This section describes the problem definition of the proposed system, the unconstrained optimization model and the constrained optimization model of the proposed system in detail.

4.3.1 Problem Formulation

Although, there is increasing use of cloud computing for scientific experiments in the form of scientific workflow execution, researchers in the scientific area have poor knowledge of which resources (i.e., VMs) they should use for their application that optimize performance with the lowest financial resource deployment cost. In that case, optimal performance means not only minimizing the makespan (execution time) but other factors like optimizing cost, availability probability and failure probability of VMs during workflow execution. Moreover, from the perspective of cloud service providers, it is difficult for them to

provide the services with users' optimal constraints with unsuitable resource request. The proposed system is intended to let such researchers focus on their subjective experiments rather than the resource selection for the experiments. At the same time, the system helps service providers to provide services within the user-specified constraints through optimal resource selection.

4.3.2 Decision Variables

The goal of this multi-objective optimization system is to find the optimal resource deployment plan for a scientific application in terms of makespan, deployment cost, virtual machine failure probability and virtual machine availability probability in a level-wise manner. Consequently, the decision variables for the system are chosen to be the number of VMs allocated to each level of scientific workflows with their respective machine instance types. The current system is focused on scientific workflows with a homogeneous structure and homogenous instance types are assigned to the leveled tasks. The optimization problem is treated as unconstrained and constrained minimization and maximization problem on two, three and four objective functions in a subsequent model development process.

4.3.3 Mathematical Models for Objective Functions

The final proposed system model encompasses the evolution of the three experiments conducted by the following mathematical models.

4.3.3.1 Bi-objective Unconstrained Optimization Model

In the bi-objective optimization model, the objective functions utilized in the proposed system are geared towards minimizing the overall makespan for workflow execution and machine instance deployment cost during workflow execution. They are expressed as fol-

lows:

$$\min t(X), \min c(X)$$

where,

$X = [x_1, x_2, \dots, x_l, \dots, x_m]$ is the number of VMs and types of VMs at each level of the DAG

$t(X)$ = total workflow execution time

$c(X)$ = total workflow machine instance deployment cost

L = total number of levels in the workflow DAG ($m = 2L$)

1. Level-wise makespan cost is calculated as follows:

$$t(X) = \sum_{k=1}^L \max_{j=1}^m T_{kj} \quad (4.1)$$

where,

T_{kj} = task execution time of j^{th} cluster at level k

m = number of clusters in level k

2. Level-wise virtual machine deployment cost

$$c(X) = \sum_{k=1}^L \max_{j=1}^m T_{kj} \times x_k \times P(x_k) \quad (4.2)$$

where,

$$P(x_k) = \begin{cases} 0.0160, & \text{if } x_{l+k} = 4 \\ 0.0039, & \text{if } x_{l+k} = 3 \\ 0.0016, & \text{if } x_{l+k} = 2 \\ 0.0004, & \text{if } x_{l+k} = 1 \end{cases} \quad (4.3)$$

x_k = number of instances allowed for level k

$P(x_k)$ = instance price for workflow level k

The possible machine instance types are small, medium, large, and x-large for this model. The deployment cost differs according to the machine instance type used during workflow optimization. $P(x_k)$ signifies the price of the machine instance type in level k specified in Hokkaido University Academic Cloud pricing. According to the machine instance type, the runtime of each task in the workflow is also different. We assume that significant cost-performance tradeoff will occur depending on the machine instance we choose during optimization.

4.3.3.2 Three-Objective Unconstrained Optimization Model

$$\min t(X), \min c(X), \min f(X)$$

where,

$f(X)$ = total workflow machine instance failure probability during execution

In scientific workflow applications, execution dependencies among tasks are important. Thus, level-wise execution resource selection is conducted in our proposed system. First, we optimize the makespan and the number of VMs at each execution level. Then, following optimization at all levels in the workflow, an optimized execution plan is identified for the overall workflow. Moreover, the deployment cost differs per machine instance type used

and the number of machine instances used during workflow optimization. According to the machine instance type, the runtime of each task in the workflow is also different. For the failure minimization case, the system uses the availability factor of the virtual machine instances provided by Amazon Web Services (AWS). The cloud model supports 25 VM instance type in this three objective optimization. No user-defined constraints are considered at the moment and automatic makespan, failure and resource usage cost optimization are performed by the system. The three objective functions for level-wise makespan cost, instance deployment cost and virtual machine instance failure are as follows:

1.

$$t(X) = \sum_{k=1}^L \max_{j=1}^m T_{kj}$$

2.

$$c(X) = \sum_{k=1}^L \max_{j=1}^m T_{kj} \times x_k \times P(x_j^k)$$

3. Level-wise virtual machine failure probability during workflow execution

$$f(X) = 1 - \left[\prod_{k=1}^L \left(1 - \prod_{j=1}^{x_k} P_j^{fail} \right) \right] \quad (4.4)$$

where,

P_j^{fail} = failure probability of the j^{th} virtual machine

4.3.3.3 Four-Objective Constrained Optimization Model

$$\min t(X), \min c(X), \min f(X), \max a(X)$$

where,

$a(X)$ = total workflow machine instance availability probability during resource deployment

Constraints are considered in the four-objective optimization model and we assume that

β, D, ϕ are the constraints for the budget, deadline and the least acceptable failure probability during optimization. The objective functions conduct the optimization on four objectives while satisfying the above constraints. β, D, ϕ values of the proposed system are calculated based on the execution time of the critical path of the scientific workflow applications with the lowest performance virtual machine instance.

The constrained objective functions of makespan, virtual machine deployment cost, virtual machine failure probability and virtual machine availability probability are as follows:

1.

$$t(X) = \sum_{k=1}^L \max_{j=1}^m T_{kj}$$

subject to: $t(X) \leq D$ (4.5)

2.

$$c(X) = \sum_{k=1}^L \max_{j=1}^m T_{kj} \times x_k \times P(x_j^k)$$

subject to: $t(X) \leq \beta$ (4.6)

3.

$$f(X) = 1 - \left[\prod_{k=1}^L \left(1 - \prod_{j=1}^{x_k} P_j^{fail} \right) \right]$$

subject to: $t(X) \leq \phi$ (4.7)

4. Level-wise virtual machine availability probability

$$a(X) = \prod_{k=1}^L \left(\prod_{i=1}^{x_k} P_i^{avail} \right) \quad (4.8)$$

where,

P_i^{avail} = availability probability of VM instance during resource deployment

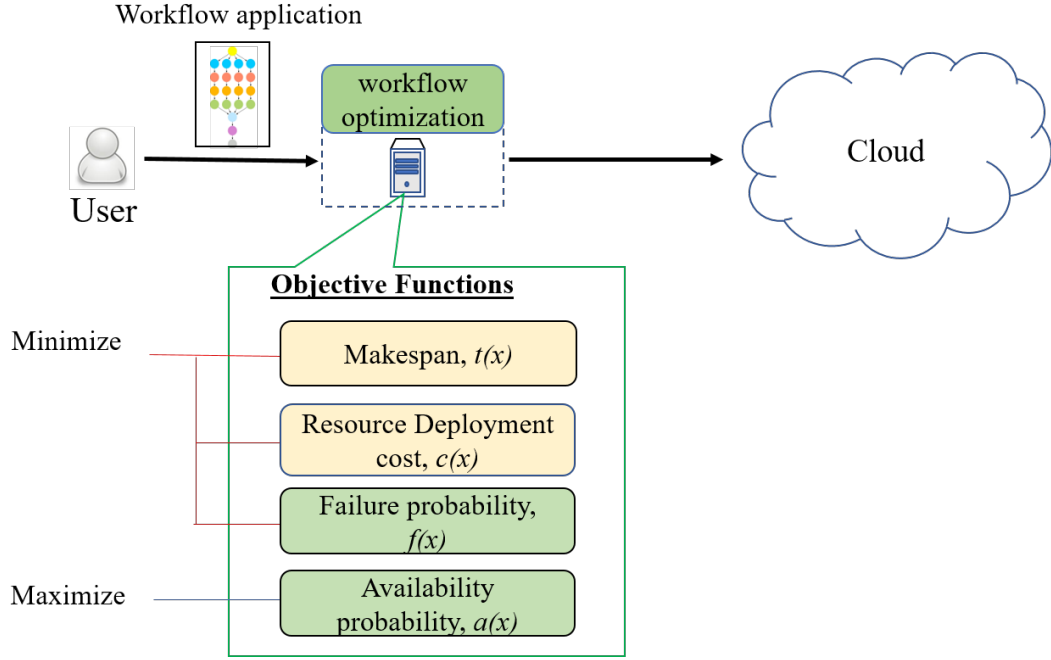


Figure 4.6: Proposed Multi-Objective Scientific Workflow Optimization System.

4.4 Optimization System

This section explains in details of the cloud system model considered in our proposed system and the idea of how workflow optimization is conducted in cloud computing for the above objective functions.

4.4.1 System Architecture

The architecture of our proposed multi-objective workflow optimization system is shown in Figure 4.6. The operation of the system is as follows. First, workflow input in the form of a DAG is given to the system for optimization and the system parses the necessary information at each level of the workflow DAG. Then, the system performs makespan minimization, virtual machine deployment cost minimization, VM failure probability minimization, and VM availability maximization based on the specific machine instance type throughout the search space. Machine instances are selected to execute all levels of a particular workflow in the cloud. Two major processes are performed during the multi-objective

optimization process: workflow parsing and multi-objective optimization.

4.4.1.1 Workflow DAX Parsing

Scientific workflows support and automate the execution of error-prone, repetitive tasks in scientific applications. It combines activities and computations to solve scientific problems. Task partitioning is important during workflow execution because it can affect the performance of the application. Three approaches to workflow partitioning before execution are reported in the literature: 1) horizontal workflow partitioning, 2) vertical workflow partitioning, and 3) arbitrary workflow partitioning.

In our proposed system, we use level-wise workflow partitioning which is a horizontal partitioning approach. We chose level-wise partitioning of structured scientific workflows because these workflows consist of different numbers of tasks with different resource requirements at each level of the workflow. The level-wise approach enables elastic provisioning of resources and load balancing at each workflow level. Level-wise partitioning of a simple DAG workflow is illustrated in Figure 4.7.

Workflow parsing retrieves task information specified in the DAG. In this step, necessary information such as total number of levels in the workflow, task name and their relevant execution time, and the maximum parallelization degree for the input workflow are parsed in order for the optimization by the system to proceed automatically. Subsequent chromosome length determination depends on the total number of levels in the workflow. Moreover, task labelling is also conducted at each level of the workflow to facilitate level-wise optimization of workflow tasks within the optimization process during workflow parsing.

4.4.1.2 Multi-Objective Optimization

Following parsing and task labelling of the workflow, the system begins the multi-objective optimization process. First, random initial chromosome parameter lists comprising the number of VMs and the type of VMs suitable for each level of the workflow are

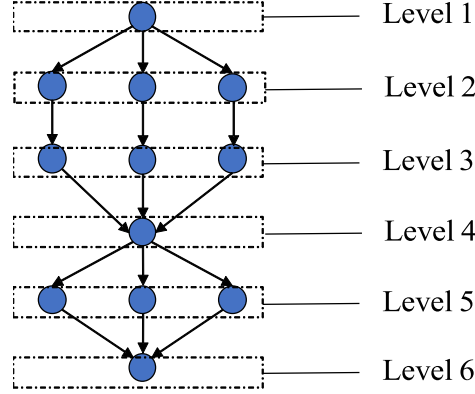


Figure 4.7: Level-Wise Workflow Partitioning.

generated to initiate the process. Next, fitness calculations are performed for the objective functions (2-objectives, 3-objectives or 4-objectives) according to the applied optimization models. Genetic operators such as selection, crossover, and mutation are then chosen to be used during evolution. Then, the evolution process proceeds until the specified termination criterion is met. At this point, the Pareto front solutions from the evolution process are retrieved and the optimized number of VMs with optimized instance type at the Pareto front for optimal makespan and optimal deployment cost for each workflow level is recommended for processing the scientific workflow applications in the cloud.

4.4.2 Cloud System Model

In our workflow optimization study, the cloud is modeled as a set of virtual machine instances, $Ins = \{ins_1, ins_2, \dots, ins_j\}$. Firstly, the 2-objective optimization was conducted with a private cloud system model that only supports four VM instance types (small, medium, large, x-large). 3-objective and 4-objective optimization concerned with a public cloud model in which each instance, ins_j can be one of 25 machine instance types as shown in Figure 4.8 and instance costs are calculated based on instance prices for Tokyo region in AWS pricing [36]. Table 4.2 shows the instance pricing used in our resource usage calculation. The performance difference of different machine instance types is specified based

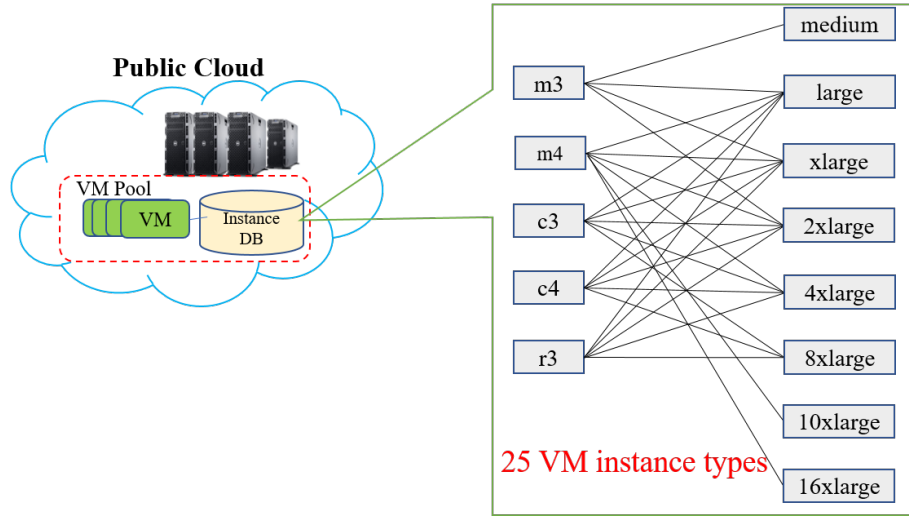


Figure 4.8: Public Cloud System Model.

on the CPU used as well as the number of CPU cores in each machine instance. We refer to the CPU performance of various instances from the PassMark benchmark [38]. Instance type m3 is assumed to be the lowest performance machine instance with the lowest price. The CPU performance differences of m4, c3, c4 and r3 instance types with respect to m3 machine instance type are assumed to be as listed in Table 4.1.

Table 4.1: Machine Instance CPU Performance

No	Instance Type	CPU Type	PassMark CPU Mark	Speedup over m3 CPU performance(%)
1	m3	Intel Xeon E5-2670 v2	14975	-
2	m4	Intel Xeon E5-2686 v4	17795	18
3	c3	Intel Xeon E5-2680 v2	16341	9
4	c4	Intel Xeon E5-2666 v3	24877	66
5	r3	Intel Xeon E5-2670 v2	14975	-

The performance speed up of each machine instance group is specified based on the number of CPU cores used in each machine instance. In addition, we also assume that all virtual machine instances share data from the same data center. Thus, data communication and data transfer costs are negligible.

Table 4.2: Machine Instance Pricing

No	Instance Type	CPU (No. of cores)	Memory (GB)	Price per second (\$)
1	m3.medium	1	3.75	0.00002667
2	m3.large	2	7.5	0.00005361
3	m3.xlarge	4	15	0.00010694
4	m3.2xlarge	8	30	0.00021389
5	m4.large	2	8	0.00004833
6	m4.xlarge	4	16	0.00009667
7	m4.2xlarge	8	32	0.00019306
8	m4.4xlarge	16	64	0.00038639
9	m4.10xlarge	40	160	0.00096583
10	m4.16xlarge	64	256	0.00154528
11	c3.large	2	3.75	0.00003556
12	c3.xlarge	4	7.5	0.00007083
13	c3.2xlarge	8	15	0.00014194
14	c3.4xlarge	16	30	0.00028361
15	c3.8xlarge	32	60	0.00056750
16	c4.large	1	3.75	0.00003694
17	c4.xlarge	4	7.5	0.00007361
18	c4.2xlarge	8	15	0.00014750
19	c4.4xlarge	16	30	0.00029472
20	c4.8xlarge	36	60	0.00058944
21	r3.large	2	15	0.00005556
22	r3.xlarge	4	30.5	0.00011083
23	r3.2xlarge	8	61	0.00022167
24	r3.4xlarge	16	122	0.00044333
25	r3.8xlarge	32	244	0.00088667

4.4.2.1 Failure of Virtual Machine Instances for Workflow Execution

The likelihood of cloud resource failures is more serious than traditional parallel systems because the cloud uses commodity servers for cost-effectiveness. Failures of cloud based resources may lead to failures of application processing. Consequently, these failures affect the performance of the workflow execution. Especially in compute intensive application cases, a failure can waste tremendous amount of time and resources [2]. Thus, it is necessary to minimize the possibility of failures in the system. . In our approach, we use balanced task clustering approach for better load balancing during workflow execution. We assume that the failure probability of a virtual machine instance per second during work-

flow execution is 3.164857×10^{-11} and calculate the failure probability of VM instances for the whole workflow during workflow execution .

4.4.2.2 Availability of Virtual Machine Instances for Instance Deployment

Availability is one of the critical aspects of Quality of Service (QoS) in the cloud. It signifies the probability that the system can complete a predetermined function under a predetermined condition. The proposed system uses the availability figure as $P_i^{avail} = 0.9999$ of AWS EC2 machine instance for computing overall availability probability of workflow during execution. Firstly, we compute task-cluster wide availability probability according to the task balanced clustering and assignment approach described in section 4.4.4. Then, we compute overall availability value based on all level-wise workflow availability probability values.

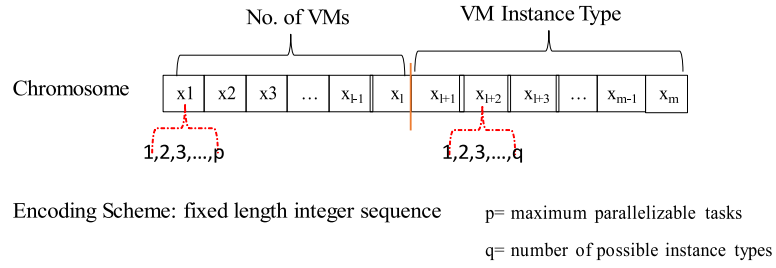


Figure 4.9: Chromosome Structure for Optimization during Workflow Execution.

4.4.3 Chromosome Representation

The chromosome of the proposed system includes genes that represent the number of VMs and the type of VMs used for each level of the workflow. The system uses a fixed-length integer-encoding scheme, one of the most common representations of chromosomes in genetic algorithms (GA). The genes in the chromosomes are in the form $\{1, p\}$ and $\{1, q\}$ with each gene representing the number of VMs for each level of workflow execution for parallelization, while p signifies the maximum number of concurrent tasks allowable in

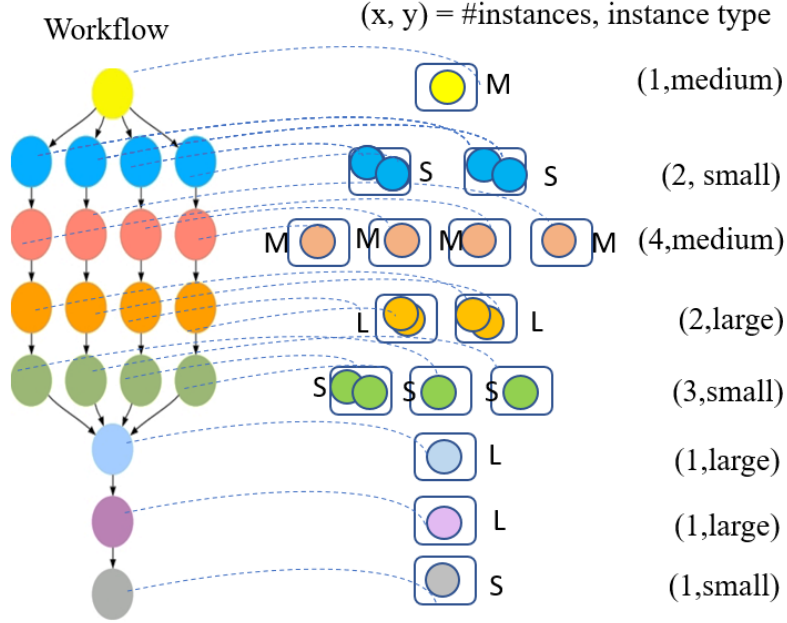


Figure 4.10: Workflow to Chromosome Mapping.

that workflow level, and q signifies the total possible number of instance types provided by the academic cloud system. A single gene represents one level and the chromosome length depends on the maximum number of levels in the workflow. The chromosome length, $2L$, where L is the highest level of the workflow DAG, varies according to the input workflow structure. Specifically, the chromosome length for a simple workflow in Figure 4.7 is ($2L=12$) where $L=6$ and the gene value ranges are $(1,p)=(1,3)$ and $(1,q)=(1,25)$ respectively. An example chromosome and workflow to chromosome mapping for the workflow optimization process is illustrated in Figure 4.9 and Figure 4.10 respectively .

4.4.4 Balanced Task Assignment

Level-wise balanced task assignment to the VMs is conducted in scientific workflow execution because the fine-grained tasks in each workflow must be processed in their dependency order. The objective of balanced task assignment is to reduce the waiting time to continue to the next level during workflow execution and to balance the loads in each machine instance at each workflow level allowing resource sharing facility. For those pur-

poses, balanced clustering of tasks at each level is considered before execution in their respective virtual machine. The system ensures that task clusters have finishing times that are as close as possible. Figure 4.11 shows the task clustering for the balanced task assignment process. There are two stages for balanced clustering: (1) task sorting and (2) balanced clustering. In task sorting, all the tasks in each level are sorted according to their runtime. Then, the task cluster size decision for each task group in the sorted task list is conducted according to the following equation:

if (number of VMs for i^{th} level= n), $NC = n(1 \leq n \leq p)$

$$C_{size_j} = \frac{T_i}{NC} + s_k, j = 1, \dots, NC, k = 1, \dots, n \quad (4.9)$$

where,

$$s_k = \begin{cases} 0, & \text{if } T_i \% NC = 0 \\ 1, & \text{if } 0 < T_i \% NC < NC \end{cases}$$

T_i = total number of tasks in i^{th} level

NC = number of task clusters

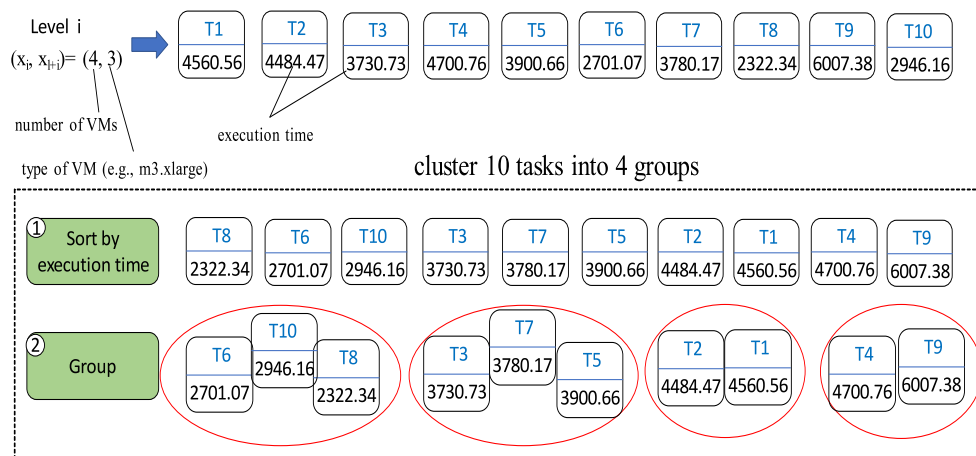


Figure 4.11: Level-Wise Balanced Task Assignment.

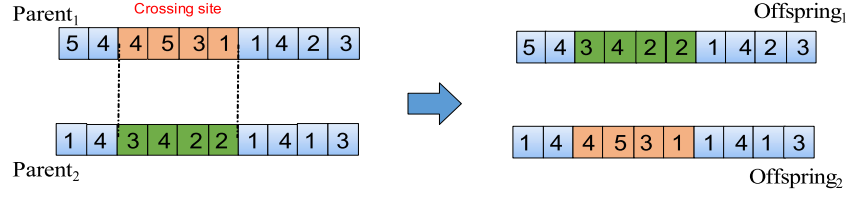


Figure 4.12: Crossover Operation.

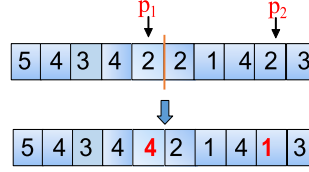


Figure 4.13: Mutation Operation.

4.4.5 Genetic Operators

Genetic operators are used to guide the multi-objective optimization algorithm towards optimal solutions to a given problem. The genes in the chromosomes represent (number of VMs, type of VMs) at each workflow level and the proposed workflow optimization processes the execution of tasks in a level-wise manner to preserve the dependencies between tasks. During optimization, the genetic operators are used to guide the multi-objective optimization algorithm towards optimal solutions to a given problem. Three main operators, selection, crossover, and mutation, collaborate with each other for successful processing. Offspring are selected via the tournament selection mechanism.

(1) Selection: The select operation gives us a straightforward way of choosing offspring for the next generation. Tournament selection is used in this system as it reduces the computational cost associated with selection. In tournament selection, random T (user-defined tournament selection size) individuals from the population are chosen, their fitness values compared, and the fittest used in the recombination process. The higher the T value, the higher the selection pressure in tournament selection.

(2) Crossover: The crossover operation takes two parent strings called $parent_1$ and $parent_2$, and generates two offspring, $offspring_1$ and $offspring_2$. The decision of

whether to perform a crossover operation on the current pair of parent chromosomes is made according to the specified crossover probability (CP). If the random probability is less than CP, a crossing site is selected and partial exchange occurs between the two parents to produce the offspring. If the probability is greater than CP, the crossing site will be the entire chromosome length, which has no effect because of the crossover operator. Two-point crossover is applied in the proposed system, as illustrated in Figure 4.12.

(3) Mutation: Mutation is important because it allows the evolutionary process to explore new potential solutions to the problem. To avoid inbreeding in a small population problem, mutation is applied with small probability (MP) after the crossover operation. The selected MP should be reasonable because if it is too high or too low, a good solution will not be found during evolution. As our chromosome structure composed of two portions with different gene value range, two random mutation points are selected from chromosome, C by the mutation operator and then replaces the gene value with the value in the range of 1, p and 1, q respectively. Example mutation operation is illustrated in Figure 4.13.

4.4.6 Penalty Function

The three penalty functions are considered for our constrained optimization of scientific workflow execution system. Penalty functions degrade solutions that do not meet constraints to leave the solutions out in the next population [31]. The penalty function of our proposed system is in the following form:

$$P_{makespan}(X) = \begin{cases} t(X) - D & \text{if } t(X) + \theta > D \\ 0, & \text{otherwise} \end{cases} \quad (4.10)$$

$$P_{cost}(X) = \begin{cases} c(X) - \beta & \text{if } c(X) + \alpha > \beta \\ 0, & \text{otherwise} \end{cases} \quad (4.11)$$

$$P_{failure}(X) = \begin{cases} f(X) - \phi & \text{if } f(X) + \gamma > \phi \\ 0, & \text{otherwise} \end{cases} \quad (4.12)$$

where $P_{makespan}(X)$, $P_{cost}(X)$, and $P_{failure}(X)$ are the penalties for violating the three objectives functions (makespan, cost and failure). D and β are the deadline and budget for the workflow application. The current D value is assumed to be the sum of task execution time in critical path of the workflow. β is the deployment cost of the workflow calculated based on the critical execution time on the highest performance machine instance in the experimental cloud system model whereas ϕ value is assumed to be 0.02337. θ , α and γ are penalty terms for the three objective functions above. θ and α are calculated by taking 0.01% of the average of makespan and cost solutions in our previous work [23].

4.5 Multi-objective Workflow Optimization Algorithm

This section presents the SPEA2, NSGA-II and NSGA-III based optimization algorithms, which are the major component of the proposed system. The system uses NSGA-II because it outperforms other contemporary MOEAs, such as PAES [13] and SPEA [6], in terms of finding a diverse set of solutions and in converging near the true Pareto-optimal set [15]. SPEA2 is used to compare the Pareto optimal solutions obtained from NSGA-II optimization. For more than 3-objective optimization, NSGA-III is used in the proposed optimization system.

Algorithm 4.1: Preprocessing

Input: Workflow DAG.

Output: Task Objects, total number of levels (L).

Step 1: Parse the workflow DAG

Step 2: Label tasks

Step 3: Assign workflow tasks to objects

Step 4: Retrieve maximum workflow level for chromosome length determination in optimization

Step 5: Retrieve maximum parallelization degree for search space determination

Algorithm 4.2: Workflow Optimization using SPEA2

Input: Task Objects, total number of levels (L), N: Population size,

G: number of generations, O: number of objectives, MP: mutation probability,

CP: crossover probability.

Output: Pareto optimal solutions, W_{opt} .

Processing

Step 1: Generate random initial population, P_0 and empty archive $\overline{P}_0 = \Phi, t = 0$.

1.1 Randomly generate integer chromosome $C_i (i = 1, 2, \dots, N)$ for $P_0, \overline{P}_0$.

(truncate if necessary)

1.2 Calculate the fitness of each random chromosome

1.3 Copy all non-dominated individuals in P_t and $\overline{P}_t$ to $\overline{P}_{t+1}$

Step 2: Evolve population ($t, P_t, \overline{P}_t, \overline{P}_{t+1}$).

2.1 Select parents using tournament selection, $\overline{P}_{t+1}$

2.2 Perform two-point dynamic crossover operation on parents with probability CP to produce offspring population.

2.3 Perform mutation operation with probability MP on random points applied to

offspring

2.4 Calculate fitness of new offspring population and update population

2.5 Use non-dominated sorting to divide into several non-domination levels

$$F_1, F_2 \cdots, F_l.$$

2.6 Update $\overline{P_t + 1}$

Step 3: Repeat Step 2 until termination condition is met.

Step 4: Output the list of optimal solutions at Pareto front, W_{opt} .

Algorithm 4.3: Workflow Optimization using NSGA-II

Input: Task Objects, total number of levels (L), N: Population size,

G: number of generations, O: number of objectives, MP: mutation probability,

CP: crossover probability.

Output: Pareto optimal solutions, W_{opt} .

Processing

Step 1: Generate random initial population, IP.

1.1 Randomly generate integer chromosome $C_i (i = 1, 2, \cdots, N)$.

1.2 Calculate the fitness of each random chromosome

1.3 Rank population P according to non-domination

Step 2: Evolve population (N, P, G, MP, CP).

2.1 Select parents using tournament selection, (P_1, P_2) .

2.2 Perform two-point dynamic crossover operation on parents (P_1, P_2) with probability CP to produce offspring population Q_t .

2.3 Perform mutation operation with probability MP on random points applied to offspring Q_t

2.4 Calculate fitness of new offspring population and update population

2.5 Use non-dominated sorting to divide $P \leftarrow P \cup \{x^C\}$ into several non-domination levels $F_1, F_2 \dots, F_l$.

2.6 Calculate Crowding Distance of all solutions

2.7 Identify the worst solution $x' \in F_l$ and set $P \leftarrow P' \setminus \{x'\}$.

Step 3: Repeat Step 2 until termination condition is met.

Step 4: Output the list of optimal solutions at Pareto front, W_{opt} .

Algorithm 4.4: Workflow Optimization using NSGA-III

Input: Task Objects, total number of levels (L), N: Population size,

G: number of generations, O: number of objectives, MP: mutation probability,

CP: crossover probability, Z^r : Reference points.

Output: Pareto optimal solutions, W_{opt} .

Processing

Step 1: Generate random initial population, IP.

1.1 Randomly generate integer chromosome $C_i (i = 1, 2, \dots, N)$.

1.2 Calculate the fitness of each random chromosome

1.3 Associate Reference points

Step 2: Evolve population (N, P, G, MP, CP).

2.1 Select parents using tournament selection, (P_1, P_2) .

2.2 Perform two-point dynamic crossover operation on parents (P_1, P_2) with probability CP to produce offspring population Q_t .

2.3 Perform mutation operation with probability MP on random points applied to offspring Q_t

2.4 Calculate fitness of new offspring population and set $R_t = P_t \cup Q_t$

2.5 Apply non-dominated sorting on R_t and find $F_1, F_2 \dots, F_l$.

```

2.6  $S_t = \{\}, i = 1$ 
2.7 While  $|S_t| \leq N$  do .
2.8  $S_t = S_t \cup F_i$ .
2.9  $i = i + 1$ 
2.10 End
2.11 If  $|S_t| = N$  do
2.12  $P_{t+1} = S_t$ ; break.
2.13 Else
2.14  $P_{t+1} = \cup_{j=1}^{l-1} F_j$ 
2.15 Normalize  $S_t$  and interpret the points of each objective
2.16 Associate each member of  $S_t$  with a reference point
2.17 Choose  $N - |P_{t+1}|$  members from  $F_l$  by niche preserving operator
2.18 End

```

Step 3: Repeat Step 2 until termination condition is met.

Step 4: Output the list of optimal solutions at Pareto front, W_{opt} .

4.6 Performance Evaluation

This section outlines the experimental environment and presents the experimental results obtained for various scientific workflow applications. The simulation experiments were conducted on an Intel core i7 Dell PC with 16 GB RAM with Pegasus workflow dataset. The system implementation was conducted using the Python programming language and the well-known DEAP framework [8].

4.6.1 Experimental Scientific Workflow Applications

To demonstrate the proposed offline workflow optimization system, the experiments were conducted using three scientific workflow DAG applications (Epigenomics, Montage, and CyberShake) from Pegasus [22] [34], with various input data sizes. Epigenomics is a CPU-bound bioinformatics workflow with eight levels of workflow tasks. Montage is a collection of programs that combines multiple shots of astronomical images to generate a custom mosaic image [15]. CyberShake is a seismology workflow application that is used by the Southern California Earthquake Center to characterize earthquake hazards in a region [5]. The structures of the applications are illustrated in Figure 4.14. Table 4.3 shows the scientific workflows used for system evaluation along with their respective input data sizes.

Table 4.3: Experimental Workflows

No	Workflow Application	Number of task nodes (Input size)			
1	Epigenomics	24	46	100	997
2	Montage	25	50	100	1000
3	CyberShake	30	50	100	1000

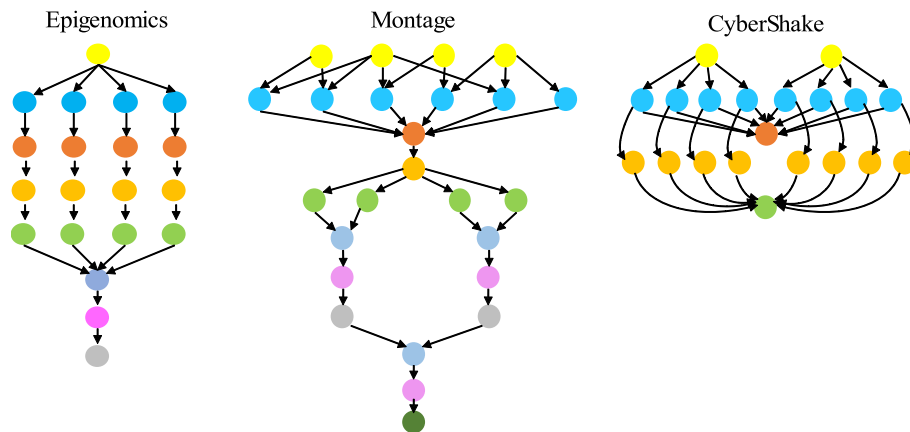


Figure 4.14: Structure of the Experimental Scientific Workflows.

4.6.2 Experimental Setup

The section describes the parameter settings were used for NSGA-II and NSGA-III algorithms for 2-objectives, 3-objectives and 4-objectives optimizations in our proposed approach, as shown in Table 4.4. Table 4.5 lists the penalty values used for 4-objective constrained workflow optimization experimentations.

Table 4.4: Algorithm Parameters

No. of Objectives	Population Size	No. of Generations	Crossover Probability	Mutation Probability
2	100	1000	0.9	0.2
3	100	500	0.7	0.0625 0.055 0.125
4	100	300	0.7	0.0625 0.055 0.125

Table 4.5: Penalty Values for Constrained Optimization

Workflow	Penalty Terms (θ, α, γ)		
Epigenomics 24	9.639245×10^{-3}	2.831180×10^{-5}	1.590083×10^{-13}
Epigenomics 46	1.592588×10^{-2}	1.157490×10^{-4}	1.951504×10^{-13}
Epigenomics 100	9.138218×10^{-2}	2.748863×10^{-3}	1.013376×10^{-12}
Montage 25	1.134733×10^{-4}	3.838866×10^{-7}	2.704810×10^{-15}
Montage 50	2.548270×10^{-4}	1.671861×10^{-6}	4.393360×10^{-15}
Montage 100	3.208890×10^{-4}	9.062853×10^{-6}	4.053620×10^{-15}
CyberShake 30	4.019950×10^{-4}	1.561386×10^{-6}	4.860720×10^{-15}
CyberShake 50	5.762490×10^{-4}	5.355098×10^{-6}	5.451440×10^{-15}
CyberShake 100	8.099970×10^{-4}	1.889069×10^{-5}	3.817360×10^{-15}

4.6.3 Experimental Analysis

The ensuing sections describe the experimental results obtained for our experiments on bi-objective, three-objective, and four-objective optimizations of scientific workflow execution in private and public cloud system models.

4.6.3.1 Experimental Results for Bi-objective Optimization

Figure 4.15 shows the optimal solutions identified by the proposed system using the bi-objective resource optimization model. The private cloud system model that supports four types of VM instances was used during the bi-objective workflow optimization experiments. The experiments were conducted with a population size of 100 evolving over 1000 generations with crossover probability 0.9 and mutation probability 0.2 using NSGA-II. Real-world workflows Epigenomics, Montage, and CyberShake were used. The experimental results indicate that the system speeded up the makespan effectively with acceptable cost tradeoff. In Epigenomics 24 node workflow, the system selected the best options with makespan and cost minimized by a factor of 12. For Montage 25 workflow, the system improved the makespan by a factor of 19 and deployment cost by a factor of 10 in the solution range. In addition, it improved the makespan time in the CyberShake 30 workflow by a maximum factor of 13 in terms of both makespan and cost. Further, the larger the input data size, the more efficient the system became. The effectiveness of the system is reflected more in the larger data size workflows, in which makespan is significantly improved with only a small increase in deployment cost.

4.6.3.2 Experimental Results for Three-objective Optimization

The current experimental cloud system model is the extended version of the previous private cloud system model that supports 25 VM instance types. Figure 4.16 to Figure 4.19 show cost-makespan tradeoff solutions identified by the proposed system using NSGA-II and SPEA2 for 3-objectives optimization. The experiments were conducted with a population size of 100 evolving over 500 generations with crossover probability 0.7 and mutation probability 0.0625, 0.055, and 0.125—in accordance with the number of levels in the given workflow. The experimental results indicate that the system reduced the makespan effectively with acceptable cost tradeoff. For failure minimization objective, the system gives only some significant effectiveness in smaller input data size workflows compared with

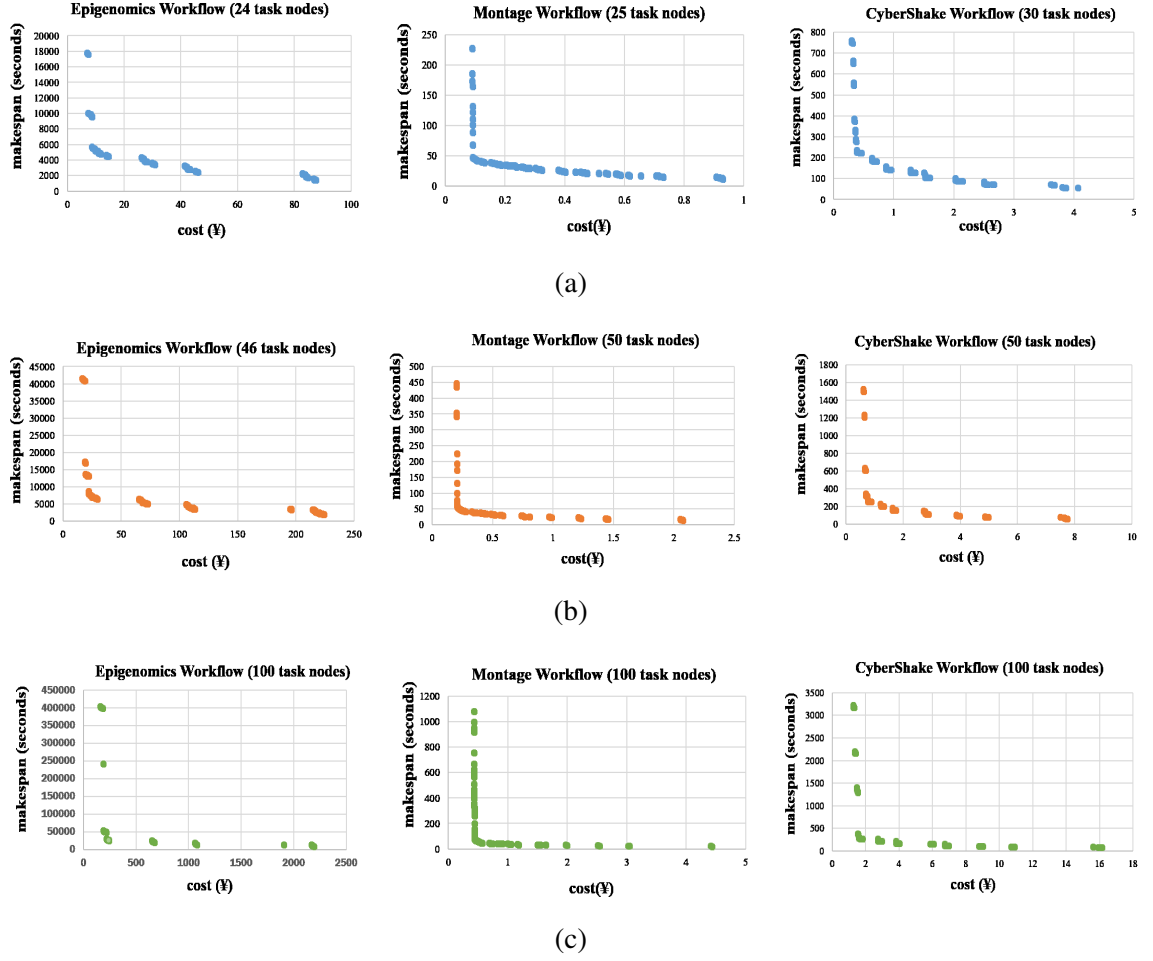


Figure 4.15: 2-Objective Cost-Makespan Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large Input Data Size using NSGA-II

other two objective functions.

Figure 4.16 shows cost-makespan tradeoff solutions for three workflows with four input data size after 500 generations. Although the optimization was conducted on three objective functions, the distribution of the solutions is presented in two dimensions (2D) as 2D provides better clarity for comparison than 3D. In the figures, it is clear that makespan and deployment cost functions are strong correlated with a negative slope. That is, the higher the makespan, the lower the deployment cost, and vice versa. As our objective functions are minimizing functions, we would like to choose the optimal solution which is as close to the X-Y origin as possible. The values in the figures also illustrate a gentle slope

in small input data size that becomes steeper with increased data size.

The scatter plots in Figure 4.17 illustrate the cost-failure tradeoff solutions. The values in the figures highlight that the two functions are not strongly correlated. Nevertheless, there is a weak correlation between these two functions for all three types of workflows (Epigenomics, Montage, and CyberShake). The current experimental input workflows are in the form of homogeneous workflow type whereas each level-wise tasks have similar characteristics; all Pareto front solutions recommend to use similar instance type but with different number of machine instances at each workflow level. Moreover, the failure probability values of all solutions are very similar with only different around 7^{th} or 8^{th} position of a fractional value. This can affect the cost-failure tradeoff solutions of the workflows to be a linear function with no correlation. For these reasons, the best optimal solution for the three objective functions should be selected based on the makespan, cost and failure respectively. In our evaluation, we identified two extreme points and one mid-point (the one closet to the origin) to be makespan best, time best and mid-point solution to find out the system effectiveness. We found out that the system improves (2.4,5.7) times for (makespan, cost) for Epigenomics case with failure 1.9394×10^{-9} , (3.0,0.17) times improvement with failure 1.1451×10^{-11} for Montage and (4.6,3.9) times improvement with failure 3.5094×10^{-13} for CyberShake respectively. Figure 4.18 and 4.19 show the experimental results using SPEA2 algorithm and the results are quite similar for all three workflows.

For the evaluation of the evolutionary algorithms, several quality indicators such as convergence, diversity, inverse generational distance (IGD) and hypervolume exist. Among these indicators, convergence and diversity values of the optimal solutions obtained from the Pareto fronts over five independent test runs for 500 generations were used to calculate the average values and standard deviations listed in Table 4.6. These quality indicator calculations were conducted according to the DEAP framework, the smaller the convergence and diversity values, the better. Convergence is a measure of the distance between the obtained non-dominated front and the true Pareto optimal front. Convergence metric, γ is

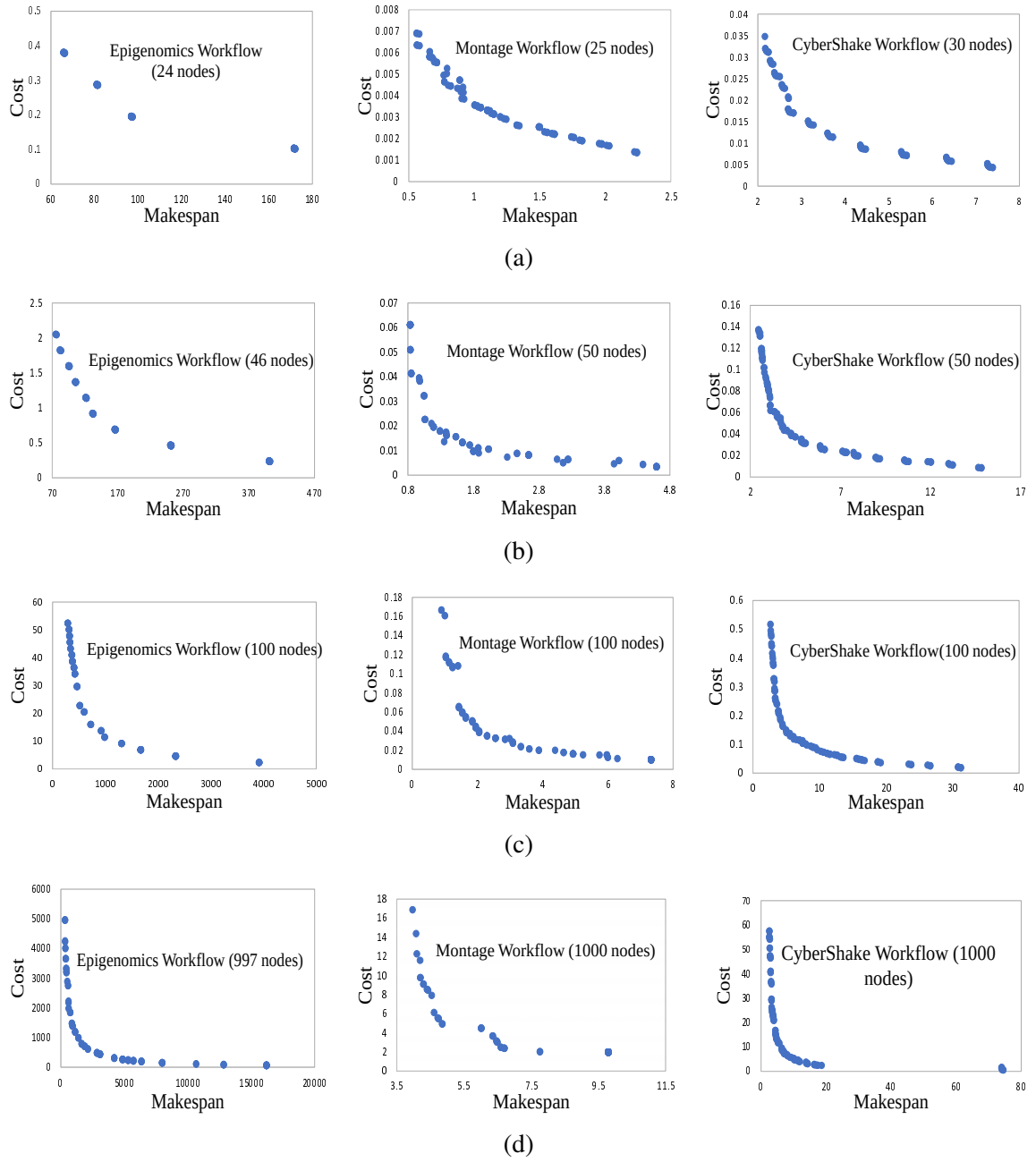


Figure 4.16: Cost_Makespan Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large input data size using NSGA-II

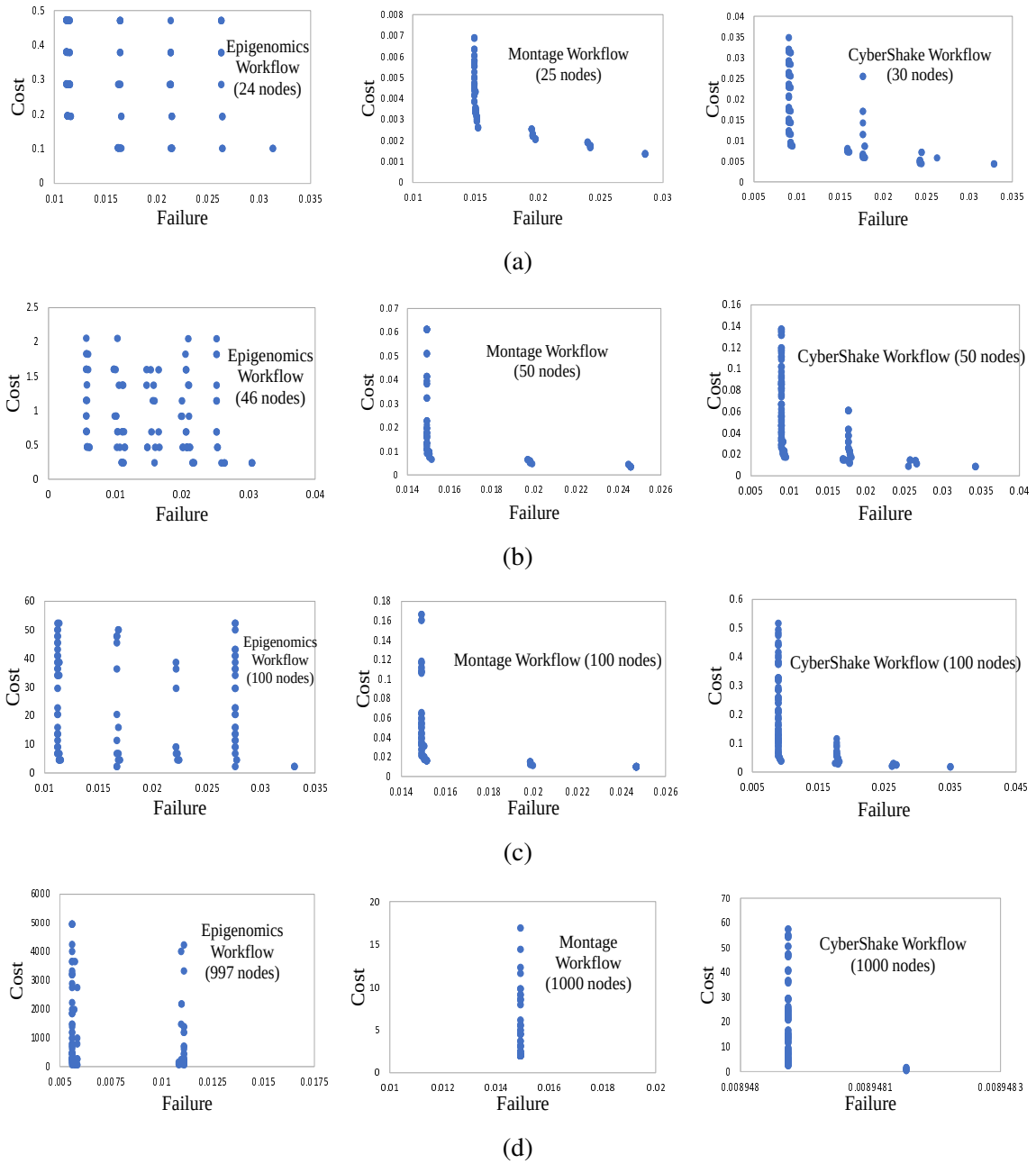


Figure 4.17: Cost.Failure Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large Input Data Size using NSGA-II

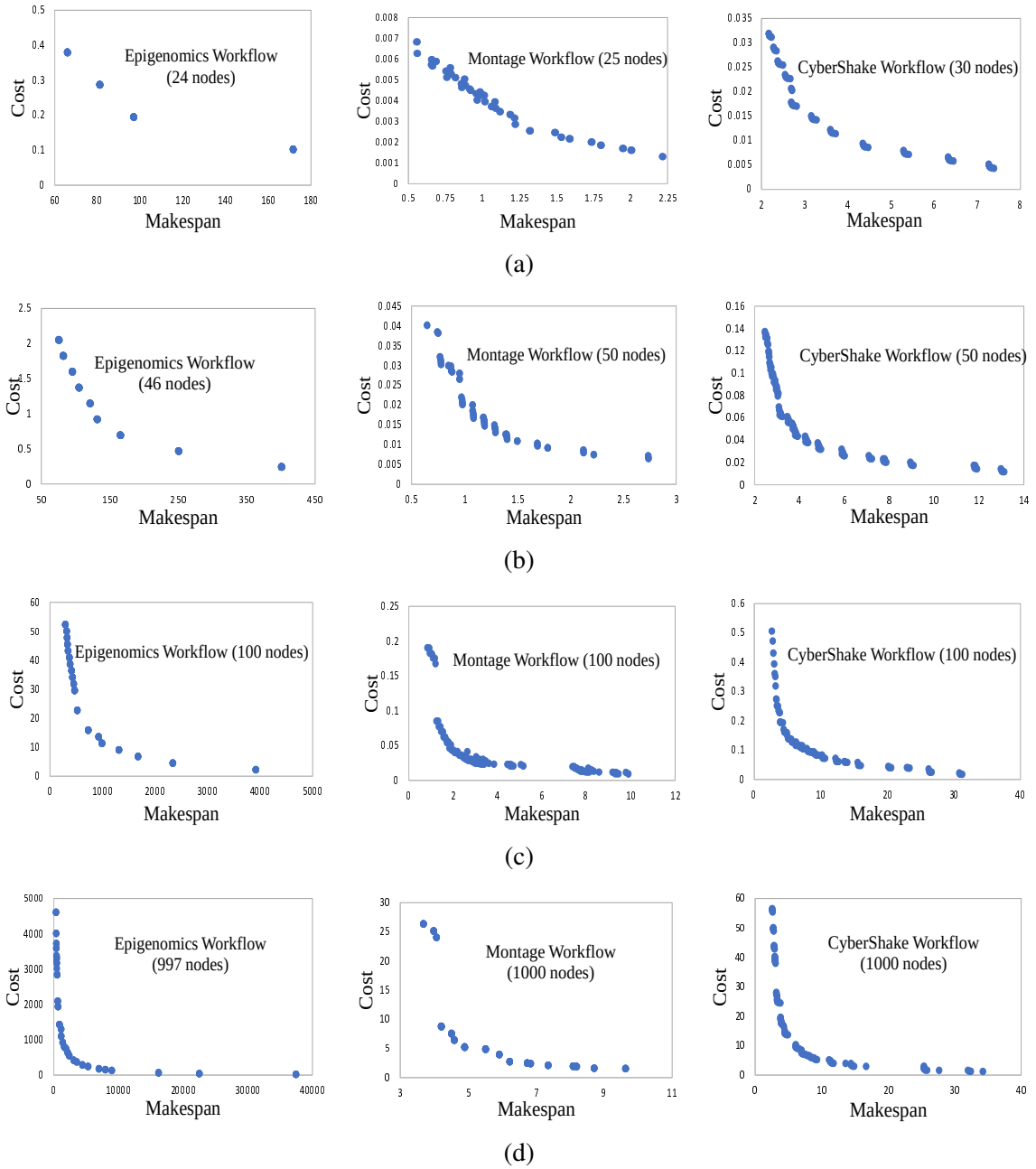


Figure 4.18: Cost_Makespan Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large Input Data Size using SPEA2

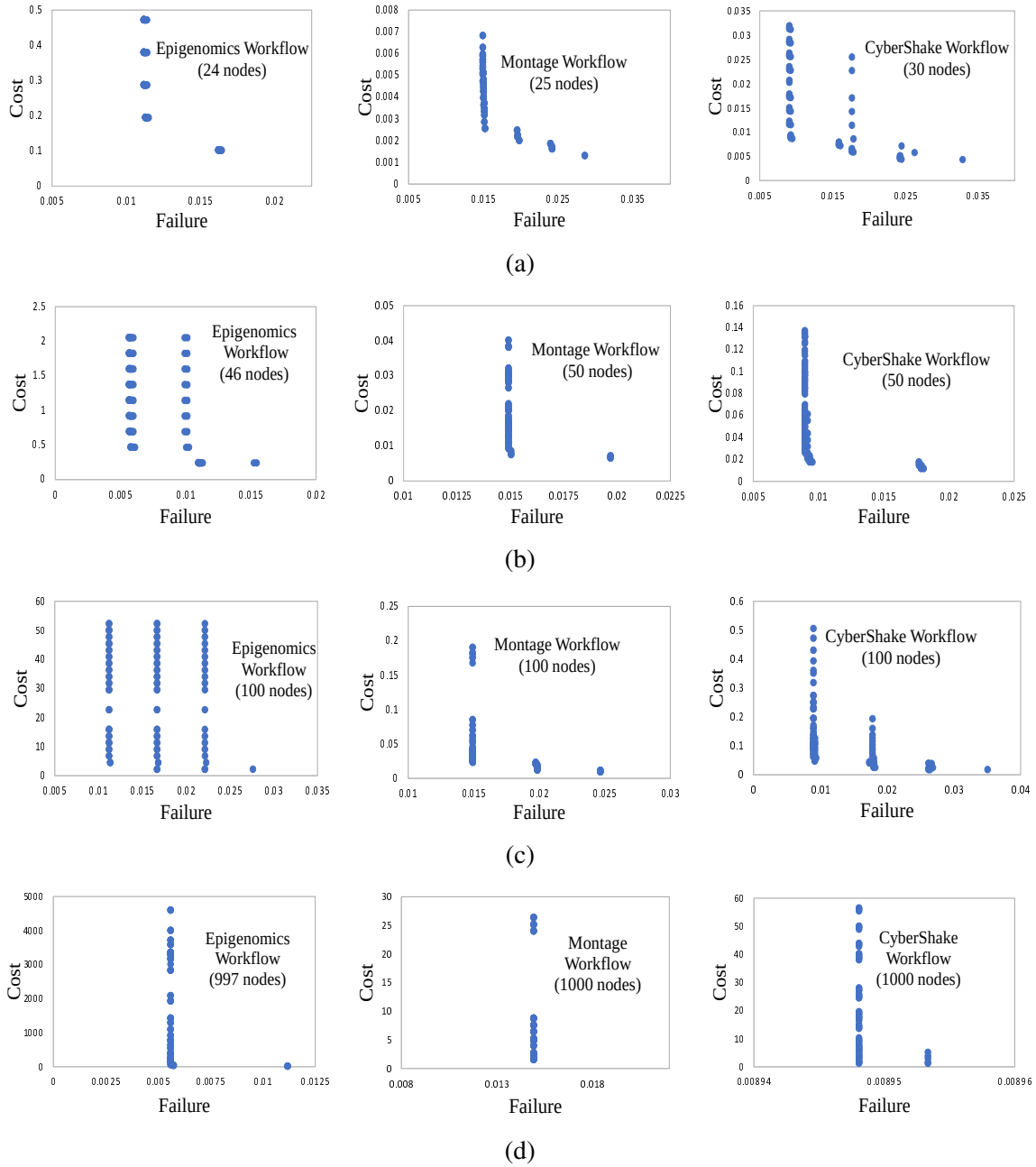


Figure 4.19: Cost.Failure Pareto Solutions a, b, c, d for Three Workflows with Small, Medium, Large and X-large Input Data Size using SPEA2

calculated by computing the minimum Euclidean distance of each obtained solution from the chosen solutions on the Pareto-optimal front and then averaging those distances [26]. The metric is calculated as below:

$$\text{Convergence, } \gamma = \frac{\sum_{i=1}^N |d_i|}{N} \quad (4.13)$$

where, d_i = Euclidean distance between the obtained solutions and the nearest member of the best non-dominated front

Diversity is a measure of the sufficiency of the obtained non-dominated solutions to represent the range of the Pareto-optimal front. The equation for diversity calculation is described in [14] as follows:

$$\text{Diversity, } \Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N - 1)\bar{d}} \quad (4.14)$$

where,

d_f, d_l = Euclidean distances between the extreme solutions and the boundary solutions

d_i = a solution on the best non-dominated front $d_i, i = 1, 2, 3, \dots, (N - 1)$

$\bar{d}$ = the average of all distances d_i

The current research work focuses on optimization of scientific workflow executions in the cloud through MOEAs such as SPEA2, and NSGA-II. According to the experimental results, the ability to identify the Pareto optimal solutions of SPEA2 and NSGA-II are virtually the same. In some cases, SPEA2 gives better convergence and diversity values than NSGA-II. However, a longer execution time is required by SPEA2 which is very important for the search process during workflow optimization. In MOEAs such as SPEA2

and NSGA-II, the most important parts of the algorithms are in fitness assignment based on Pareto-domination: domination counts, non-dominated sorting, and identification of the non-dominated solutions. These processes can be done within $O(MN^2)$, where M is the number of objectives and N is the population size in both SPEA2 and NSGA-II. The time complexities of SPEA2 and NSGA-II are beyond the scope of this paper, but interested readers can refer to Jessen [18] for further details on the time complexities of MOEAs.

Table 4.6: Convergence, Diversity and Execution Time for SPEA2 and NSGA-II

Workflow	Algo-rithm	Avg. Conv. ($\pm\sigma$)	Avg. Div. ($\pm\sigma$)	Avg. Exec. (sec) ($\pm\sigma$)
Epigenomics_24	SPEA2	0.0210 (± 0.0103)	1.8596 (± 0.0348)	409.3682 (± 46.8026)
	NSGA-II	0.0140 (± 0.0036)	1.8728 (± 0.0140)	47.7937 (± 11.3710)
Montage_25	SPEA2	0.0008 (± 0.0020)	1.5249 (± 0.1714)	377.6937 (± 28.2203)
	NSGA-II	0.0039 (± 0.0022)	1.5211 (± 0.1776)	44.0162 (± 2.9298)
CyberShake_30	SPEA2	0.0086 (± 0.0007)	1.3504 (± 0.0020)	433.8772 (± 12.7594)
	NSGA-II	0.0076 (± 0.0014)	1.3540 (± 0.0040)	40.6471 (± 9.8157)
Epigenomics_46	SPEA2	0.1301 (± 0.0803)	1.7365 (± 0.0594)	410.9416 (± 85.4199)
	NSGA-II	0.0690 (± 0.0144)	1.7658 (± 0.0165)	48.2626 (± 6.2828)
Montage_50	SPEA2	0.0029 (± 0.0051)	1.5091 (± 0.1759)	410.9416 (± 85.4199)
	NSGA-II	0.0057 (± 0.0037)	1.5958 (± 0.2331)	47.1161 (± 9.0593)
CyberShake_50	SPEA2	0.0318 (± 0.0046)	1.2235 (± 0.0954)	411.7136 (± 15.8681)
	NSGA-II	0.0234 (± 0.0034)	1.2973 (± 0.0474)	44.9424 (± 9.5771)
Epigenomics_100	SPEA2	0.0996 (± 0.0856)	1.7282 (± 0.0141)	409.5439 (± 24.9837)
	NSGA-II	0.1371 (± 0.2226)	1.7210 (± 0.0129)	59.5994 (± 8.7646)
Montage_100	SPEA2	0.0070 (± 0.0062)	1.3036 (± 0.2084)	382.4262 (± 24.2236)
	NSGA-II	0.0162 (± 0.0137)	1.4850 (± 0.1835)	54.1009 (± 3.8788)
CyberShake_100	SPEA2	0.0703 (± 0.0128)	1.0123 (± 0.1563)	402.9489 (± 11.9935)
	NSGA-II	0.0472 (± 0.0131)	1.2627 (± 0.1153)	50.3276 (± 9.6360)
Epigenomics_997	SPEA2	2.4378 (± 2.3017)	1.5537 (± 0.0589)	545.1640 (± 17.8064)
	NSGA-II	10.075 (± 4.431)	1.5512 (± 0.0568)	193.7554 (± 10.1521)
Montage_1000	SPEA2	0.0118 (± 0.0210)	1.5282 (± 0.1355)	560.4173 (± 82.2385)
	NSGA-II	0.0972 (± 0.0972)	1.5802 (± 0.1836)	216.3977 (± 22.6453)
CyberShake_1000	SPEA2	0.3411 (± 0.1213)	1.2386 (± 0.1502)	561.0543 (± 56.9944)
	NSGA-II	0.2325 (± 0.0526)	1.3233 (± 0.1178)	164.8595 (± 6.7466)

4.6.3.3 Experimental Results for Four-objective Optimization

Figures. 4.20 shows the optimal solutions identified by the proposed system for 4 objective optimizations. The experiments were conducted with a population size of 100 evolving over 300 generations with crossover probability 0.7 and mutation probability 0.0625, 0.055 and 0.125 in accordance with the number of levels in the workflows (Epigenomics, Montage, and CyberShake) using NSGA-III. The following 3D images illustrate three objectives while specifying the fourth objective values as color bar. The experiments give Pareto solutions that are identified during the optimization process for all three workflows on three different input data sizes. Further details of the experimental results are discussed in the ensuing chapter.

4.7 Related Work and Discussion

The use of cloud in scientific experiments is a popular alternative because transferring large volumes of data to processing nodes is impractical in this era of continuously increasing Big Data. However, the right virtual machine instance selection for a specific application is a challenging problem for the cloud users. Thus, optimization system approaches that easily facilitate optimized cloud resource selection are actively being researched.

Workflows represent complex distributed scientific computations that comprise many tasks, the execution of which may require many complex modules and software [28]. Workflow optimization of scientific applications has become an interesting research area in this era of continuously increasing Big Data. Thus, various single-objective and multi-objective optimization strategies have been proposed. The proposed single-objective optimization strategies primarily focused on developing effective scheduling strategies to reduce makespan.

Tanaka et al. [19] proposed a data-aware scheduling strategy that reduces makespan by minimizing data movement between cluster nodes. However, their strategy can only

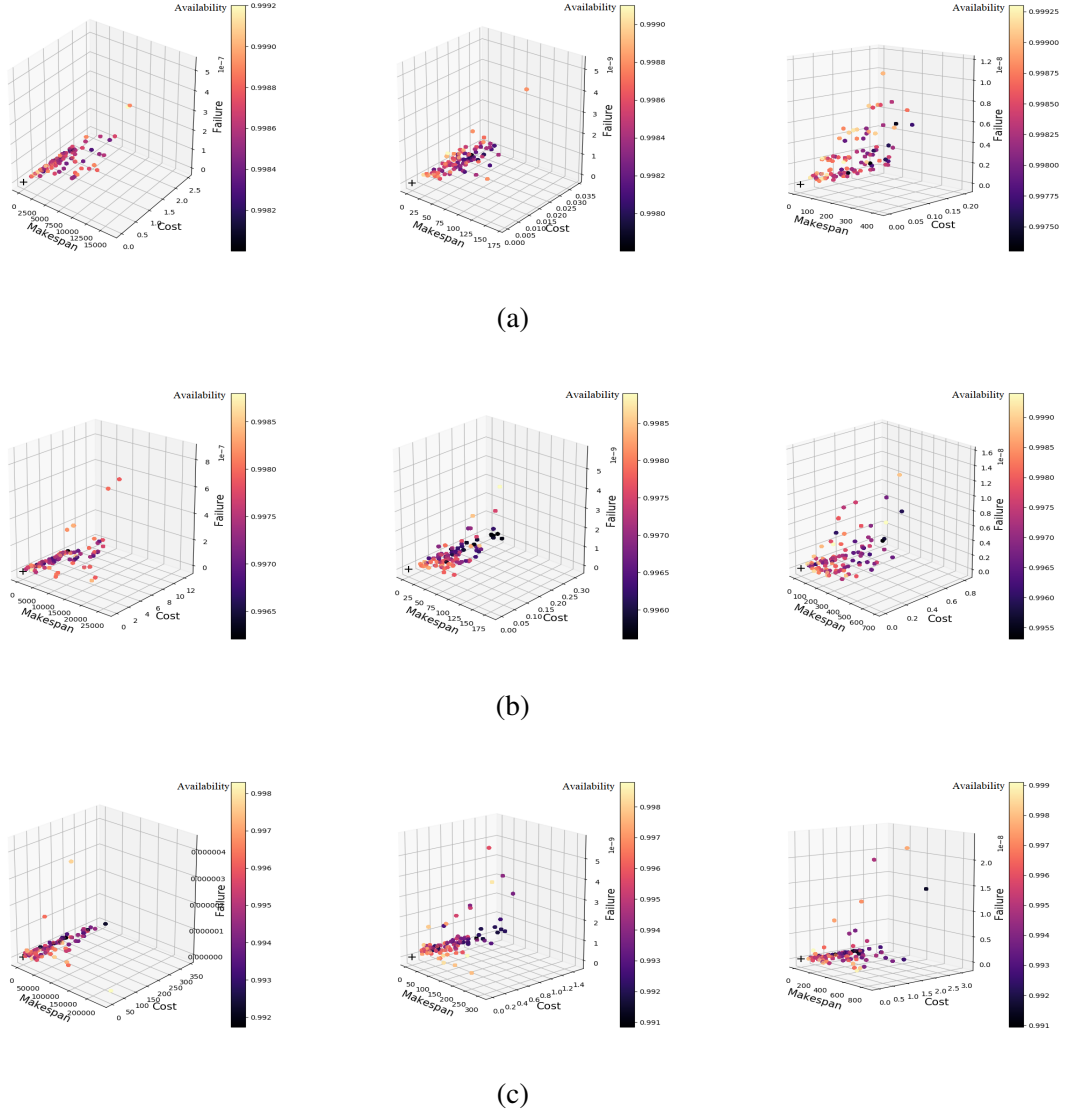


Figure 4.20: Pareto Solutions a, b, and c for Three Workflows with Small, Medium, and Large Input Data Size using NSGA-III

operate with homogeneous resources during workflow execution and is not suitable for use in cloud environments. Chen et al. [28] proposed a workflow scheduling strategy that uses horizontal or vertical task clustering techniques to reduce the schedule overhead during workflow execution. Yan et al. [3] proposed a novel probability evaluation-based scheduling algorithm to address the problem of workflow deadline guarantee. They showed that their approach is effective in terms of adaptability and predictability under deadline con-

constraints, but improvements in deadline distribution is still necessary. Kaur et al. [21] proposed workflow scheduling in clouds using the augmented shuffled frog leaping algorithm (ASFLA). They improved the original shuffled frog leaping algorithm (SFLA) to speed up the overall execution time of the workflow. Experiments conducted showed significant reduction in overall workflow execution time, but resource usage cost was not accounted for in the system. Poola et al. [4] presented a resource scheduling algorithm for workflow execution on heterogeneous cloud resources along with makespan and cost minimization. However, the robustness and effectiveness of their proposed system depend on an increased budget. Zhu et al. [30] proposed a multi-objective workflow scheduling approach for cloud infrastructure. In their proposed approach, they enhance the evolution process of their optimization by introducing two novel crossover and mutation operators. They conducted experiments based on actual pricing and resource parameters on Amazon EC2, but did not consider task dependencies during workflow executions and simply assumed that all tasks can be executed in parallel. Zhang et al. [17] proposed bi-objective workflow optimization on energy consumption and reliability in heterogeneous computing systems. Single-objective and multi-objective workflow execution optimization has been conducted on Cluster, Grid, and Cloud Infrastructure in recent years. Single-objective optimization approaches are focused on minimizing the workflow makespan through scheduling [3] [11] [17] and task clustering approach [28]. However, single-objective optimization is not sufficient for data intensive workflows. Our proposed method is focused on multi-objective workflow execution optimization in the cloud infrastructure. Although similar work [4] [21] has been carried out on cloud infrastructure, the approaches employed have limitations in terms of resource elasticity and task dependencies. Moreover, previous studies do not cover important issues associated with VM instances such as VM failure probability. Our proposed approach addresses these issues as follows:

1. Randomized search-based resource selection to better fit resource usage

2. Level-wise task processing approach for task dependencies and resource elasticity
3. Balanced Task Clustering for better load balancing and resource sharing
4. Optimized workflow execution on three-objectives (including cloud SLA related objectives)

Throughout our research, first we conducted unconstrained multi-optimization then we extend the system to include constraints during optimization of four objective functions, i.e., minimization of workflow makespan, instance resource cost, probability of instance failures and maximization of instance availability probability without violating the task dependencies of the workflow in the cloud infrastructure. In this way, the proposed system helps with the selection of the right resources through multi-objective optimization during workflow executions that benefits both stakeholders (users and cloud service providers).

4.8 Conclusion

Many workflow optimization studies have been conducted on grid computing systems as well as in the cloud. However, a cost-effective alternative for scientific workflow executions with pay-per-use paradigm to provide services with the best performance and the cheapest virtual machine deployment cost is still challenging. In this study, we addressed this problem by modeling the workflow optimization problem as a multi-objective optimization problem using NSGA-II and NSGA-III in the cloud.

To solve the multi-objective workflow optimization problem, we used a fixed-length encoding scheme to represent a random number of VMs that can be used in each level of the workflow. Based on the random initial chromosome population, the evaluation process was conducted, and the respective fitness values of the chromosomes calculated. We evaluated the proposed approach on three real-world workflows available from the ongoing Pegasus research project [34]. Our experimental results show that the system can speed up

makespan with an acceptable cost tradeoff. The results also highlight that optimization depends on the structure of the workflow, i.e., the more parallelized tasks there are in the level, the more the makespan time can be reduced. Our current work is focused on constrained workflow execution optimization without violating the dependency and load balancing of the machine instances during workflow execution.

Bibliography

- [1] B. Ludashcer, M.Weske, T.McPhillips and S. Bowers. *Scientific Workflows: Business as Usual?*. Proc. 7th International Conference on Business Process Management, BPM 2009, pp. 31-47, Springer 2009.
- [2] C. Chitra. *Performance comparison of multi-objective evolutionary algorithms for QoS routing problems in computer networks*. Thesis Dissertation, 2011.
- [3] C. Yan, H. Luo, Z. Hu, X. Li and Y. Zhang. *Deadline Guarantee Enhanced Scheduling of Scientific Workflow Applications in Grid*. Journal of Computers, vol.8, No. 4, April 2013.
- [4] D. Poola, S. K. Garg, R. Buyya, Y. Yang and K. Ramamohanarao. *Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds*. IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), pp. 858-865, 2014.
- [5] D. A. Prathibha, B. Latha and G. Sumathi. *EFFICIENT SCHEDULING OF WORKFLOW IN CLOUD ENVIRONMENT USING BILLING MODEL AWARE TASK CLUSTERING* Journal of Theoretical and Applied Information Technology, vol. 65, no. 3, July 2014.
- [6] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and*

Applications Doctoral dissertation ETH 13398, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

- [7] E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, May 2001.
- [8] F. Fortin, F. D. Rainville, M. Gardner, M. Parizeau and C. Gagne. *DEAP: A Python Framework for Evolutionary Algorithms*. GECCO '12 Proceedings of the 14th annual conference companion on Genetic and evolutionary computation, Pages 85-92, ACM, 2012.
- [9] H. Cui, Y. Li, X. Liu, N. Ansari and Y. Liu. *Cloud service reliability modelling and optimal task scheduling*. Journal of IET Communications 2017, Vol. 11, iss 2, pp. 161-167, The Institution of Engineering and Technology, 2016.
- [10] H. Jain and K. Deb. *An Improved Adaptive Approach for Elitist Nondominated Sorting Genetic Algorithm for Many-Objective Optimization* Proc. Of the International Conference on Evolutionary Multi-Criterion Optimization, pp 307-321, 2013.
- [11] H. Khajemohammadi. A. Fanian and T. Aaron Gulliver. *Efficient Workflow Scheduling for Grid Computing Using a Leveled Multi-Objective Genetic Algorithm*. J Grid Computing, Springer, 2014.
- [12] J. Knowles and D. Corne. *The Pareto archive evolution strategy: A new baseline algorithm for multi-objective optimization* The Congress of Evolutionary Computation, pp. 98-105.
- [13] J. Meena, M. Kumar and M. Vardhan. *Cost Effective Genetic Algorithm for Workflow Scheduling in Cloud under Deadline Constraint* IEEE Transactions and Content Mining, vol 4, pp 5065- 5082, 2016.

- [14] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. *A fast and elitist multi-objective genetic algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation, vol. 2, Issue 6, pp. 182-197, 2002.
- [15] K. Deb, L. Thiele, M. Laumann, and E. Zitzler. *Scalable test problems for evolutionary multi-objective optimization*. In Evolutionary Multiobjective Optimization, Theoretical Advances and Applications, A. Abraham, L. Jain and R. Goldberg, Eds. Springer, Heidelberg, 2005, pp. 105-145.
- [16] K. Keahey and M. Parashar. *Enabling on-demand service via cloud computing*. Proc. IEEE Cloud Computing, vol:1, issue: 1, pp. 21-27, 2014.
- [17] L. Zhang, K. Li, C. Li and K. Li. *Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems*. Information Sciences (2016), Elsevier, 2016.
- [18] M. T. Jessen. *Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms*. IEEE Transactions on Evolutionary Computation, vol.7, no.5, IEEE, 2003.
- [19] M. Tanaka, O. Tatebe. *Workflow Scheduling to Minimize Data Movement Using Multi-Constraint Graph Partitioning* Proc. 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '12, ACM, 2012,pp. 65-72.
- [20] N. Srinivas and K. Deb. *Multiobjective function optimization using non-dominated sorting genetic algorithms*. Evol.Compu., vol.2, no.3, pp.221-248, 1995.
- [21] P. Kaur and S. Mehta. *Resource Provisioning and Work flow Scheduling in Clouds using Augmented Shuffled Frog Leaping Algorithm*. Journal of Parallel and Distributed Computing, ScienceDirect, vol. 101, pp. 41-50, November 2016.

- [22] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su and K.Vahi. *Characterization of Scientific Workflows*. 2008 Third Workshop on Workflows in Support of Large-Scale Science, IEEE, 2008.
- [23] S. Chand and M. Wagner. *Evolutionary many-objective optimization: A quick-start guide* Surveys in Operations Research and Management Science, vol. 20, no.2, pp. 35-42, 2015
- [24] S. G. Ahmad, C. S. Liew, M. M. Rafique and E. U. Munir. *Optimization of data-intensive workflows in stream-based data processing models* Journal of Supercomputers, Springer, pp. 3901-3923, 2017.
- [25] S. Holl, D. Garijo, K. Belhajjame, O. Zimmermann, R. D. Giovanni, and M. Obst. *On specifying and sharing scientific workflow optimization results using research objects* Proc. 8th Workshop on Workflows in Support of Large-Scale Science, ACM, 2013, pp. 28-37.
- [26] S. Kushwaha. *Multiobjective optimization of cluster measures in Microarray Cancer data using Genetic Algorithm Based Fuzzy Clustering*. Dissertation, Department of Computer Science and Engineering, National Institute of Technology Rourkela, India, 2013.
- [27] S. Navaz and U. Anasari. *An evolutionary algorithm in grid scheduling by multi-objective optimization using variants of NSGA*, International Journal of Scientific and Research Publications, vol. 2, Issue 9, 2012.
- [28] W. Chen and E. Deelman. *WorkflowSim: A toolkit for simulating scientific workflows in distributed environments*. IEEE 8th International Conference on E-Science (e-Science), October 2012.
- [29] W. Mkaouer, M. Kessentini, A. Shaout, P. Koligheu, K. Deb and A. Ouni. *Many-*

- Objective Software Remodularization Using NSGA-III*. ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 24, Issue 3, May 2015.
- [30] Z. Zhu and G. Zhang. *Evolutionary multi-objective workflow scheduling in cloud*. IEEE Transactions on parallel and distributed Systems, vol. 27, no. 5, May 2016.
- [31] D.Simon. *Evolutionary Optimization Algorithms*. USA, Wiley Press, 2013
- [32] G. Scherp. *A Framework for Model Driven Scientific Workflow Engineering*. Dissertation, Kiel Computer Science Series, University of Oldenburg, Germany, 2013.
- [33] P. Mell and T. Grance. *The NIST definition of cloud computing*. Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech., 2011.
- [34] Pegasus Workflow Generator.
<http://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator>. Accessed November 2016.
- [35] Amazon Cloud Service.
<https://aws.amazon.com/ec2/instance-types/>. Accessed June 2017.
- [36] Hokkaido University Academic Cloud System.
<http://www.hucc.hokudai.ac.jp/hosting-server.html>. Accessed November 2016.
- [37] PassMark CPU Benchmark.
https://www.umed.pl/zp/pliki/1141_2.pdf. Accessed May 2017.

Chapter 5

Discussion

This dissertation dealt with two major forms of Big Data applications optimization. The first is Hadoop configuration optimization by conducting parameter tuning using steady-state Non-dominated Sorting Genetic Algorithm ssNSGA-II. The second is scientific workflow application execution optimization in the cloud computing infrastructure using SPEA2, NSGA-II, and NSGA-III. This chapter presents our research findings obtained during our associated experimentations.

5.1 Hadoop Configuration Optimization

Hadoop configuration optimization is very challenging because of the complexity of its framework. Further, optimized Hadoop parameter configuration settings depend significantly on the performance of MapReduce applications in the cluster. Although much research has been conducted on Hadoop parameters configuration optimization, configuring its resource setting parameters to minimize the execution time of MapReduce jobs in clusters still the subject of much research. In addition, determining the type of machine instances that should be used to minimize the resource usage cost for executing applications in clusters is also difficult. We addressed these problems by optimizing the instance re-

source usage and execution time of MapReduce tasks using the multi-objective steady-state Non-dominated Sorting Genetic Algorithm II (ssNSGA-II) approach. In this approach, the instance resource usage cost of MapReduce tasks is calculated based on the cost of machine instance types and the number of machine instances in the Hadoop cluster. The optimized configuration is identified by selecting an optimal setting that satisfies two objective functions associated with instance resource usage and execution time minimization, from Pareto-optimal front solutions. We conducted our proposed methodology verification on Hibench Benchmark with four types of workload. According to the result, terasort, sort and wordcount workloads showed that, all the parameter settings on these workloads have a significant effect on the objective functions as the points in the scatter plot are sparse for each machine instance type. In contrast, the results on pagerank had overlap points, indicating that YARN and YARN related parameters have greater effect on this type of workload. The experiments were executed on a real Hadoop cluster with cloud virtual machine instances. The experiments required restarting of the cluster with new parameters in each evaluation run with long experimentation time to calculate the fitness function for each chromosome. According to our research, in order to identify the optimal Hadoop parameters, approximately 9 hours was required for sort workload, 6 hours for terasort workload, 9 hours for wordcount workload and 14.5 hours for pagerank workload application. We conducted 150 evaluations for bi-objective optimization system because of the costly experimental iterations of Hadoop Parameter optimization. The experiments were conducted using 2.98 GB input for Micro-Benchmark and 5000 pages for the pagerank benchmark program. Although, we could not obtain the Pareto solution for x-large machine instances, our system was able to identify Pareto solutions for large, medium and small machine instances. The results indicate that large and medium instance type clusters can give suitable tradeoff between the two objective functions. The proposed bi-objective Hadoop parameter optimization was conducted with static Hadoop cluster size and dynamic cluster sizing is out of consideration of this research.

5.2 Scientific Workflow Execution Optimization

Cloud computing in the field of scientific applications such as scientific big data processing and, Big Data analytics has become popular because of its service-oriented model that provides a pool of abstracted, virtualized, dynamically scalable computing resources and services on demand over the Internet. However, resource selection to make the right choice of instances for a certain application of interest is a challenging problem for researchers. In addition, providing services with optimal performance at the lowest financial resource deployment cost based on user's resource selection is quite challenging for cloud service providers. Consequently, it is necessary to develop an optimization system that can provide benefits to both users and service providers. This issue motivated this research to conduct scientific workflow optimization as a constrained multi-objective optimization problem in the cloud infrastructure. Our review of the literature showed that there were limitations associated with task control dependencies, resource sharing, and load balancing. To alleviate these limitations, the proposed approach realizes optimization using a balanced task clustering approach in executing scientific workflow applications. Moreover, most of the related work do not cover important factors related with virtual machine instances that can significantly affect the cloud service level agreement (SLA). Thus, in our approach, we proposed a new mathematical model for constrained multi-objective resource optimization of scientific workflows in the cloud. The details of our model have been discussed in the previous chapter. The model effectiveness was verified using SPEA2, NSGA-II, and NSGA-III for three models. For the 2-objective optimization case, we proposed an optimization model and experimentations were conducted using a private cloud system model. The private cloud system was modeled by referencing the Hokkaido University academic cloud system, which supports four kinds of VM instances (small, medium, large and ex-large). The Pareto-optimal solutions obtained are shown in Figure 5.1 in which we selected some solutions such as extreme point Pareto solutions, one of the near optimal solutions,

to identify the model effectiveness. Then, we calculated the cost-makespan differences of the extreme point Pareto solutions with the selected near-optimal solution closest to the origin. We conducted these evaluations for the Epigenomics, Montage and CyberShake workflow applications. We found that our model is highly effective because the efficiency in any cost-makespan objective is always better than the amount of tradeoff. For example, according to Figure 5.1, choosing the near-optimal solutions (cost, makespan) with (¥242.396, 25148.16s), we obtained 9 times cost efficiency or 16 times time efficiency by taking small tradeoffs from the extreme point solutions. The situation holds for all three workflow applications in our experiment as shown in Figures 5.2 and 5.3, respectively.

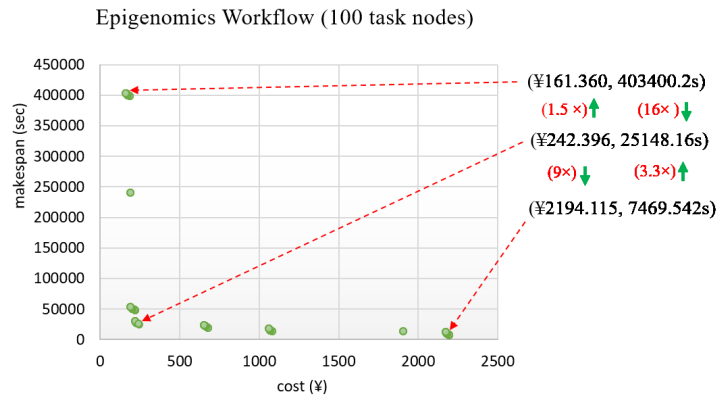


Figure 5.1: Cost-Makespan Pareto Solutions for Epigenomics Workflow on Private Cloud System.

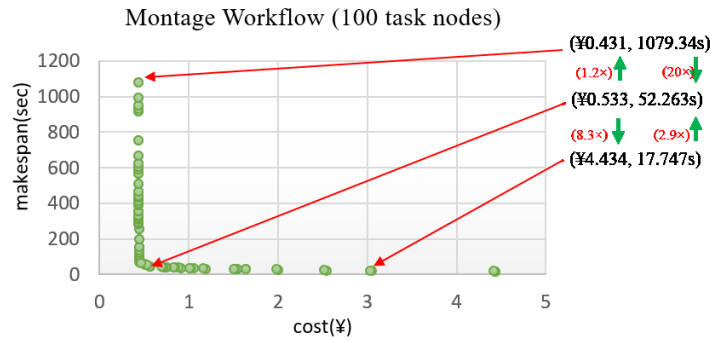


Figure 5.2: Cost-Makespan Pareto Solutions for Montage Workflow on Private Cloud System.

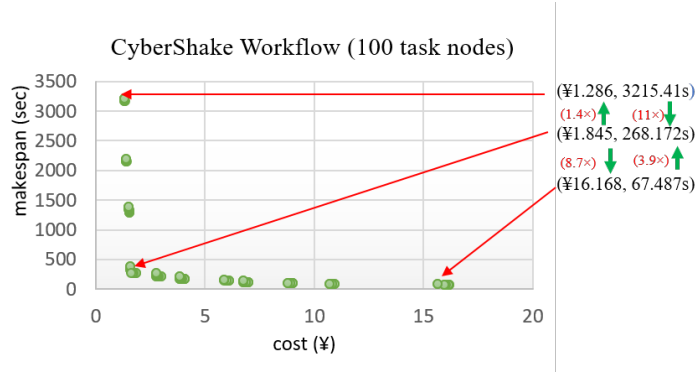


Figure 5.3: Cost-Makespan Pareto Solutions for CyberShake Workflow on Private Cloud System.

The public cloud system model that supports 25 VM types was used in 3-objective and 4-objective optimization of scientific workflow optimization experiments. According to the 3-objective optimization experimental results, the proposed optimization model can obtain Pareto solutions for cost-makespan cases. However, for the cost-failure case, we identified that these two functions only have a weak-correlation in all experimental workflows. Although some confirmation is necessary for weakly correlated function cases, the experimental results show a significant efficiency in terms of cost reduction and time saving with (5.7, 2.4) times for Epigenomics workflow application, (0.17, 3.06) times for Montage workflow application, and (3.9, 4.6) times improvement for CyberShake workflow applications compared with the extreme point solutions from the Pareto Front for 100 task nodes after the optimization process. The evaluations of those results are shown in Figures 5.4-5.6. For the cost-failure case, even though there are some outliers in the result, the optimal resource setting can be selected through the red-boundary regions shown in Figure 5.7-5.9.

From the 4-objective optimization experimental results, the optimal cost-makespan solution should be the one that is as close to the X,Y origin as possible because the two objective functions are minimization functions. The objective functions for availability and failureprobability are two contrasting functions (maximize and minimize). The shape of the Pareto solutions for different input data sizes of the same workflow type is similar for all

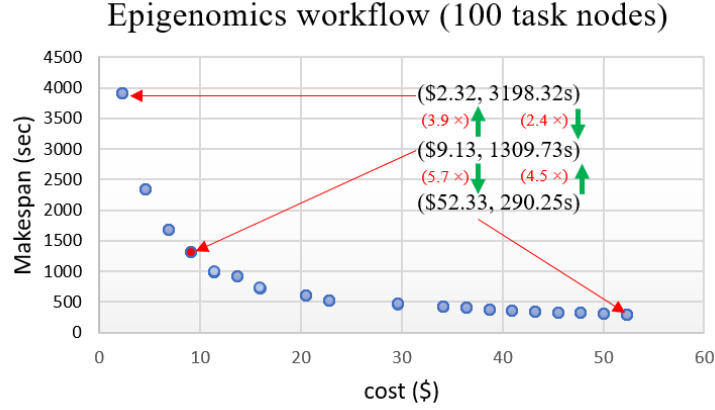


Figure 5.4: Cost-Makespan Pareto Solutions for Epigenomics Workflow on Public Cloud System.

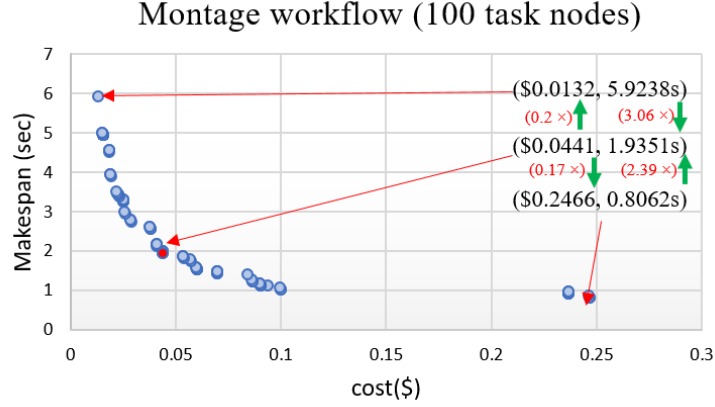


Figure 5.5: Cost-Makespan Pareto Solutions for Montage Workflow on Public Cloud System.

Epigenomics, Montage, and CyberShake. The optimal solution for the four objective functions will be lower than the specified D, β, ϕ values for all cases. The system constrains the boundary of failures, ϕ to be not more than 0.02337; thus, all optimal solutions have fewer failures with high availability probability within the specified deadline and budget during workflow execution. We extracted cost-makespan Pareto solutions from the 4-objective functions results for better readability as shown in Figures 5.10-5.12. Some mid-point Pareto solutions for the three workflows cases are described in their respective tables (Tables 5.1 - 5.12) for 4-objective optimization. According to the experimental results, the

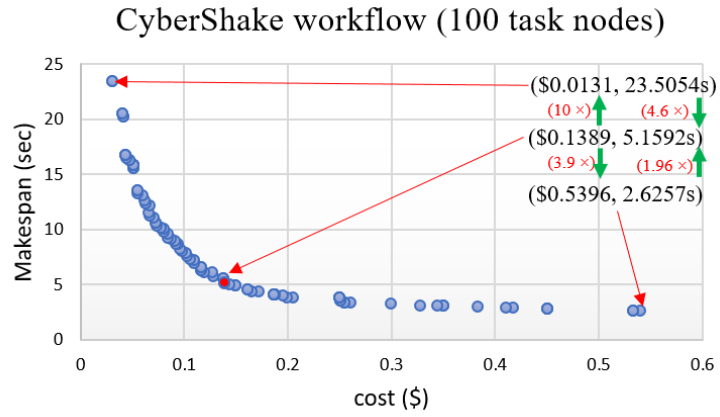


Figure 5.6: Cost-Makespan Pareto Solutions for CyberShake Workflow on Public Cloud System.

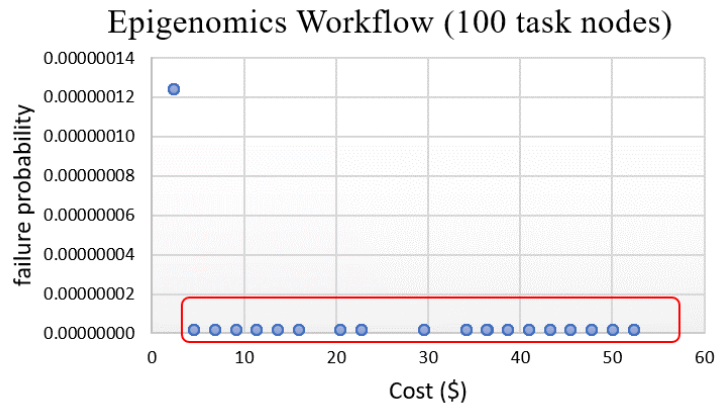


Figure 5.7: Cost-Failure Pareto Solutions for Epigenomics Workflow on Public Cloud System.

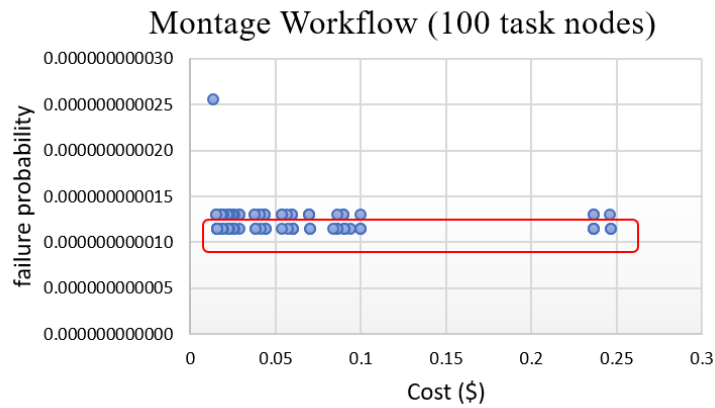


Figure 5.8: Cost-Failure Pareto Solutions for Montage Workflow on Public Cloud System.

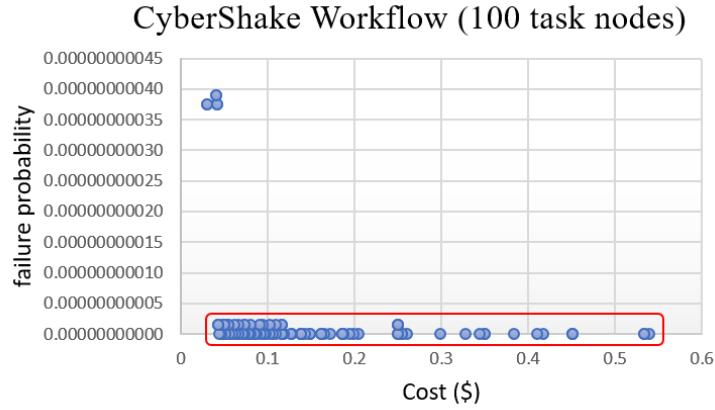


Figure 5.9: Cost-Failure Pareto Solutions for CyberShake Workflow on Public Cloud System.

lower the number of workflow levels, the better the failure probability during execution. The experimental results also highlight that our proposed optimization model is able to identify the optimal resource setting for each input application for 4-objective functions such as cost, makespan, failure probability and availability probability during workflow execution. Offline resource optimization of scientific workflow applications considering SLA in the cloud is conducted but real-time resource optimization is out of consideration of this research.

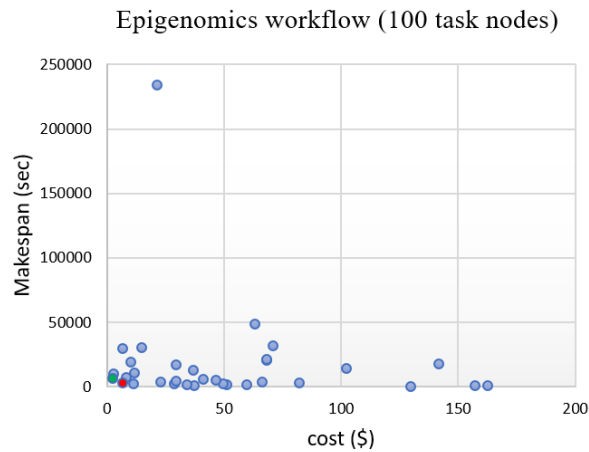


Figure 5.10: 4-Objective, Cost-Makespan Pareto Solutions for Epigenomics Workflow.

Table 5.1: Resource Setting for Epigenomics Workflow (Red Point Solution)

Level No.	Machine Type	Number of Virtual Machines
1	c4.4xlarge	1
2	m3.large	23
3	m3.medium	8
4	c4.xlarge	2
5	c4.8xlarge	3
6	m3.large	1
7	c3.xlarge	1
8	c4.xlarge	1

Table 5.2: Optimized 4-objective Values for Epigenomics Workflow (Red Point Solution)

Cost (\$)	Makespan (sec)	Failure Probability	Availability Probability
6.8977	2334.7681	0.0000000227	0.9960077901

Table 5.3: Resource Setting for Epigenomics Workflow (Green Point Solution)

Level No.	Machine Type	Number of Virtual Machines
1	c3.large	1
2	c3.2xlarge	17
3	m4.4xlarge	1
4	m4.2xlarge	23
5	c4.8xlarge	1
6	m3.large	1
7	c3.xlarge	1
8	m3.xlarge	1

Table 5.4: Optimized 4-objective Values for Epigenomics Workflow (Green Point Solution)

Cost (\$)	Makespan (sec)	Failure Probability	Availability Probability
2.5141	5971.53778	0.0000001892	0.9954103348

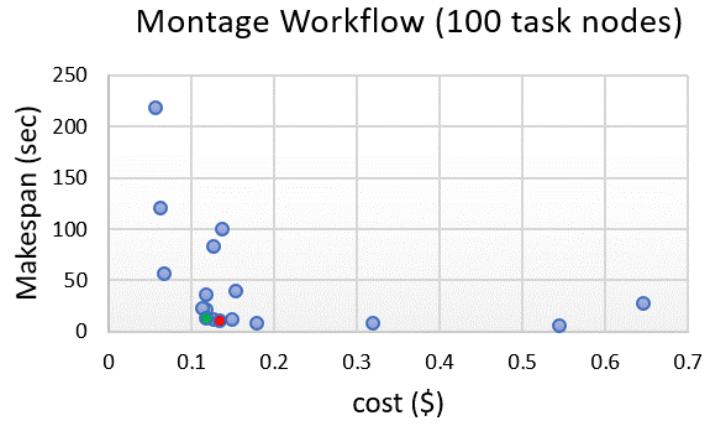


Figure 5.11: 4-Objective, Cost-Makespan Pareto Solutions for Montage Workflow.

Table 5.5: Resource Setting for Montage Workflow (Red Point Solution)

Level No.	Machine Type	Number of Virtual Machines
1	c4.4xlarge	13
2	c4.8xlarge	24
3	c4.xlarge	1
4	r3.xlarge	1
5	r3.4xlarge	5
6	c3.xlarge	1
7	c4.xlarge	1
8	c3.4xlarge	1
9	c3.xlarge	1

Table 5.6: Optimized 4-Objective Values for Montage Workflow (Red Point Solution)

Cost (\$)	Makespan (sec)	Failure Probability	Availability Probability
0.1350	10.1594	0.0000000002	0.9952112627

Table 5.7: Resource Setting for Montage Workflow (Green Point Solution)

Level No.	Machine Type	Number of Virtual Machines
1	m4.10xlarge	2
2	c4.4xlarge	17
3	c4.4xlarge	1
4	m4.xlarge	1
5	c3.8xlarge	12
6	c4.8xlarge	1
7	c3.xlarge	1
8	c3.large	1
9	r3.xlarge	1

Table 5.8: Optimized 4-Objective Values for Montage Workflow (Green Point Solution)

Cost (\$)	Makespan (sec)	Failure Probability	Availability Probability
0.1185	12.9820	0.0000000003	0.9963066522

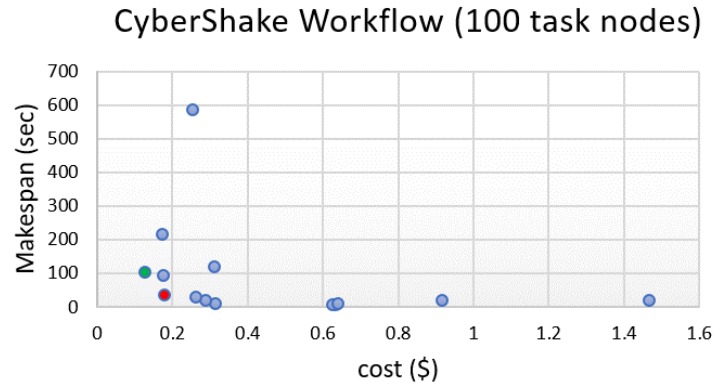


Figure 5.12: 4-Objective, Cost-Makespan Pareto solutions for CyberShake Workflow.

Table 5.9: Resource Setting for CyberShake Workflow (Red Point Solution)

Level No.	Machine Type	Number of Virtual Machines
1	c4.4xlarge	4
2	c4.8xlarge	22
3	c3.8xlarge	1
4	c3.2xlarge	42
5	m3.xlarge	1

Table 5.10: Optimized 4-Objective Values for CyberShake Workflow (Red Point Solution)

Cost (\$)	Makespan (sec)	Failure Probability	Availability Probability
0.1357	10.7083	0.0	0.9930240954

Table 5.11: Resource Setting for CyberShake Workflow (Green Point Solution)

Level No.	Machine Type	Number of Virtual Machines
1	c4.8xlarge	6
2	c4.xlarge	3
3	m3.medium	1
4	m3.medium	30
5	c3.large	1

Table 5.12: Optimized 4-Objective Values for CyberShake Workflow (Green Point Solution)

Cost (\$)	Makespan (sec)	Failure Probability	Availability Probability
0.1295	102.3708	0.0	0.9959081894

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Resource optimization of Big Data application is a critical issue in current advanced computing infrastructures such as Hadoop and the cloud. This research was conducted with the objective of improving the efficiency of Big Data application in these two technologies. The contributions of this thesis are as follows. First, Hadoop becomes a popular option for executing Big Data applications because of its fault tolerance, scalability and cost saving. However, parameter configuration is a key factor and it is very difficult to identify the parameters that significantly affect the performance of an application. Moreover, identifying application-oriented optimized machine instances for Hadoop cluster setup is very difficult. To tackle the above issues, we proposed an offline Hadoop configuration optimization methodology that optimizes the execution time and instance cluster setup cost using ssNSGA-II. The experimental results verify that our methodology improves the executions of Big Data applications in Hadoop infrastructures. Dynamic cluster resizing represents possible future direction of this Hadoop parameter optimization research.

Second, utilization of cloud systems in scientific computing has gained prominence throughout the years because of the cloud's scalability and flexibility. However, machine

instance resource selection for scientific workflow applications with the best performance, the cheapest deployment cost and minimum failure probability is still challenging in the cloud. Without proper resource selection, user dissatisfaction (owing to high costs and unreliable result data) can affect service providers' business. In this study, we addressed this problem by modeling the scientific workflow optimization problem as unconstrained optimization problems using SPEA2 and NSGA-II in the cloud. During workflow optimization, a fixed-length integer chromosome was used to represent a random number of VMs that can be deployed at each level of the workflow. The optimization process was conducted based on the random initial chromosome population and the respective fitness values of the chromosomes calculated. Our experimental results obtained on three real-world workflows available from the ongoing Pegasus research project show that the proposed system reduces makespan with an acceptable cost tradeoff and instance failure in the 3-objective optimization case. The results also indicated that the two objective functions (makespan and instance cost) are strongly related with each other, whereas failure minimization objective values are weakly related or have no relation with the other two functions in some cases. In the later experiments, we addressed the cloud resource selection problem with deadline, budget and failure probability constraints by modeling the workflow optimization problem as a constrained multi-objective optimization problem using NSGA-III. Four-objective optimization was conducted with three constrained minimization objectives and one maximization objective. Our work focused on offline workflow execution optimization without violating the constraints, task dependencies and load balancing of the machine instances during workflow execution. Implementation of the proposed model on actual commercial cloud infrastructures will be future direction of this research. In addition, real-time multi-objective optimization of workflow execution can further extend of this proposed optimization system.

Author's Publications

1. Phyto Thandar Thant, Courtney Powell, Martin Schlueter, and Masaharu Munetomo. *Multi-Objective Level-Wise Scientific Workflow Optimization in IaaS Public Cloud Environment*. Scientific Programming, Hindawi (2017) (Accepted on 2017/10/25) (IF: 0.224)
2. Phyto Thandar Thant, Courtney Powell, Martin Schlueter, and Masaharu Munetomo. *Constrained Multi-Objective Scientific Workflow Execution Optimization with NSGA-III in the Cloud*. International Journal of Computer Science and Information Security, IJCSIS (2017) (Accepted on 2017/10/18) (IF: 0.553)
3. Phyto Thandar Thant, Courtney Powell, Martin Schlueter, and Masaharu Munetomo. *A Level-Wise Load Balanced Scientific Workflow Execution Optimization using NSGA-II*. Proceedings of 17th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (IEEE/ACM CCGRID 2017) (the 2nd International Workshop on Distributed Big Data Management (DBDM 2017)), pp. 882-889 (2017)
4. Phyto Thandar Thant, Courtney Powell, Akiyoshi Sugiki, and Masaharu Munetomo. *Multi-Objective Hadoop Configuration Optimization using Steady-State NSGA-II*. Proceedings of the 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems and 2016 17th International Symposium on Advanced Intelligent Systems (SCIS&ISIS 2016), pp. 293-298 (2016)
5. Courtney Powell, Masaharu Munetomo, and Phyto Thandar Thant. *Evaluation of Three Steady-State NSGA-III Offspring Selection Schemes for Many-Objective Optimization*. Proceedings of the 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems and 2016 17th International Symposium on Advanced Intelligent Systems (SCIS&ISIS 2016), pp. 161-171 (2016)

6. Phyto Thandar Thant, Akiyoshi Sugiki, and Masaharu Munetomo. *Parameter Optimization Framework on Apache Hadoop using Steady-State Genetic Algorithm* 情報処理学会研究報告, Vol.2015-MPS-105, No.16, pp.1-2 (2015)
7. Phyto Thandar Thant, Courtney Powell, Martin Schlueter, and Masaharu Munetomo. *A Level-wise Dependency Aware Workflow Execution Optimization using NSGA-II*. 進化計算シンポジウム **2016** (2016.12.11)