

Performance Modeling of Large-Scale Parallel-Distributed Processing for Cloud Environment

Tsuguhito Hirai

Performance Modeling of Large-Scale Parallel-Distributed Processing for Cloud Environment

by

Tsuguhito Hirai

Doctoral Dissertation

submitted in partial fulfillment of
the requirement for the degree of

**Doctor of Informatics
(Systems Science)**

Department of Systems Science
Graduate School of Informatics
Kyoto University
Japan



March, 2018

Preface

In recent years, cloud computing has become a new paradigm for utilizing computing resources, and its emergence has been bringing great benefits to us. In cloud computing, the most successful application framework is parallel-distributed processing, in which an enormous task is split into a number of subtasks and those are processed independently on a cluster of machines referred to as workers. With parallel-distributed processing, huge data can be processed in a relatively short time. In this processing, however, the task completes when all the subtasks have finished, and thus subtasks which take longer for the execution result in delay in the processing of the task. This performance degradation is referred to as the issue of stragglers, and becomes more serious as the number of subtasks increases.

There are two well-known methods to alleviate the issue of stragglers: task replication and checkpointing. In task replication, each subtask is replicated, and the original subtask and its copies are assigned to multiple workers separately. The processing of the subtask ends when either the original subtask or its copy is completed. Task replication mitigates the impact of ill-conditioned workers. On the other hand, in checkpointing, the progress of the processing is periodically saved as a checkpoint during a period of the processing of a subtask. When worker failure occurs, the subtask processed by the failed worker is resumed by another worker from the latest checkpoint. Checkpointing prevents the failed subtask executing from the beginning of the processing.

This dissertation considers the impact of stragglers and the effect of task replication and checkpointing on the performance of parallel-distributed processing through analytical models and their evaluations. In Chapter 1, we provide a brief introduction of parallel-distributed processing on cloud computing. Chapter 2 investigates the impact of stragglers, modeling the task-scheduling server as a single-server queue with a number of workers. In Chapter 3, we consider the efficiency of task-replication scheduling in terms of the number of replications. To this end, the mean value and standard deviation of the processing time of a task are approximately derived with extreme value theory. Chapter 4 evaluates the effect of checkpointing method on the processing time of a task, and approximately derives explicit expressions for the optimal number of checkpoints which achieves the shortest processing time of a task. Fi-

nally, we conclude the dissertation in Chapter 5.

In this dissertation, some analytical models and their evaluation methods are developed to investigate the performance of parallel-distributed processing. The author believes that these results are useful for further optimization of system configurations on parallel-distributed processing, and hopes that they will be helpful for future research in this field.

Tsuguhito Hirai
March, 2018

Acknowledgment

First and foremost, I would like to express my profound appreciation to my advisor, Professor Yutaka Takahashi at the Kyoto College of Graduate Studies for Informatics for his patient and continuous support. His various and valuable advice guided me to complete this dissertation. I am tremendously indebted to Professor Shoji Kasahara at Nara Institute of Science and Technology for his insightful and constructive suggestions. His numerous and fruitful instruction significantly improved my writing skills. I wish to show my sincere gratitude to Associate Professor Hiroyuki Masuyama for his precise and helpful advice. He provided me with inspiring and precious ideas for analysis.

I would like to greatly thank Professor Toshiyuki Tanaka and Professor Nobuo Yamashita for their participation as members of the dissertation committee. Their sophisticated comments considerably improved this dissertation.

I am deeply grateful to all the members of Takahashi Laboratory for meaningful discussions with them. Thanks to them, I enjoyed the daily life in the laboratory. I desire to express special thanks to secretaries of the laboratory, Ms. Mieko Aso and Ms. Noriko Yurugi for their considerable help. They took care of troublesome procedures for me.

My cordial gratitude is due to my friends for their warm encouragement. Chatting with them cures my heart and gave me energy when I was in difficulty.

Last but not least, my heartfelt appreciation goes to my parents, Akio and Hisako, brothers, Yuji and Tomoaki, and relatives for their tremendous support. Without their indispensable help, I could never be here and reach the goal of this long and winding journey.

Contents

1	Introduction	1
1.1	Overview of Cloud Computing	1
1.2	Overview of Large-Scale Parallel-Distributed Processing	4
1.3	Outline of the Dissertation	6
2	Performance Degradation by the Issue of Stragglers	11
2.1	Introduction	11
2.2	Analytical Models for Two Scheduling Policies	13
2.3	Analysis	15
2.3.1	Performance Measures	15
2.3.2	Special Cases for the Worker-Processing Time Distribution	16
2.4	Numerical Examples	17
2.4.1	Model Validation	18
2.4.2	Impact of Stragglers	20
2.4.3	Efficiency of Backup-Task Scheduling	24
2.5	Summary	28
3	Performance Analysis of Task Replication	31
3.1	Introduction	31
3.2	Related Work	32
3.3	Analytical Model	34
3.4	Analysis of Performance Measures	36
3.4.1	Exact Expressions for the First and Second Moments of the Task-Processing Time	37
3.4.2	Asymptotic Formulas for the First and Second Moments of the Task-Processing Time	38
3.4.3	Exact Expressions for the Mean Total Processing Time	41
3.5	Numerical Examples	42

3.5.1	Verification of the Proposed Approximations	43
3.5.2	Optimal Number of Replications	43
3.5.3	Efficiency of Task Replication	45
3.5.4	Effect of the Starting Time of Task Replication	53
3.6	Summary	54
4	Performance Optimization of Checkpointing	57
4.1	Introduction	57
4.2	Related Work	59
4.3	Analytical Model	59
4.3.1	Model Descriptions of Large-Scale Parallel-Distributed Processing . .	60
4.3.2	Model Descriptions of the Processing of a Subtask with Checkpointing	60
4.4	Analysis	62
4.4.1	Worker Failure Probability	62
4.4.2	Processing Delay	63
4.4.3	Mean Task-Processing Time	64
4.4.4	Optimal Number of Checkpoints	66
4.5	Numerical Examples	67
4.5.1	Approximation Accuracy of the Mean Task-Processing Time	68
4.5.2	Usefulness of the Proposed Approximation for the Optimal Number of Checkpoints	71
4.5.3	Assumption Validation of the Analytical Model	72
4.6	Summary	75
5	Conclusion	79
A	Preliminary Analysis Results	81
A.1	Maximum Domain of Attraction of a Gumbel Distribution	82
A.2	Maximum Domain of Attraction of a Fréchet Distribution	82
	Bibliography	83
	Copyright Notice	89

List of Figures

1.1	Service layers of cloud computing.	2
1.2	Deployment scopes of cloud computing.	3
1.3	Execution overview of MapReduce.	6
1.4	Overview of the issue of stragglers.	7
1.5	Overview of task replication.	8
1.6	Overview of checkpointing.	9
2.1	Normal processing model (Model N).	14
2.2	Backup-task processing model (Model B).	15
2.3	Analytical model for large-scale parallel-distributed processing.	16
2.4	Tail of the worker-processing time distribution.	18
2.5	Mean response time for Model B and simulation in the case of a Pareto distribution ($\beta = 2.007$).	20
2.6	Mean total processing time for Model B and simulation in the case of a Pareto distribution ($\beta = 2.007$).	21
2.7	Mean total processing time for Model B and simulation in the case of a Weibull distribution ($\alpha = 0.2500$).	21
2.8	Maximum throughput for Model N in the case of a Weibull distribution. . . .	22
2.9	Maximum throughput for Model N in the case of a Pareto distribution. . . .	22
2.10	Mean response time for Model N in the case of a Weibull distribution. . . .	23
2.11	Mean response time for Model N in the case of a Pareto distribution. . . .	23
2.12	Ratio of the maximum throughput for Model B to that for Model N in the case of a Weibull distribution.	25
2.13	Ratio of the maximum throughput for Model B to that for Model N in the case of a Pareto distribution.	25
2.14	Ratio of the mean response time for Model B to that for Model N in the case of a Weibull distribution.	26

2.15	Ratio of the mean response time for Model B to that for Model N in the case of a Pareto distribution.	26
2.16	Ratio of the mean total processing time for Model B to that for Model N in the case of a Weibull distribution.	27
2.17	Ratio of the mean total processing time for Model B to that for Model N in the case of a Pareto distribution.	27
3.1	Examples of assigning subtasks.	35
3.2	Tail of the worker-processing time distribution in the case of a large coefficient of variation.	44
3.3	Tail of the worker-processing time distribution in the case of a small coefficient of variation.	44
3.4	Relative error of the mean task-processing time in the case of a hyper-exponential distribution ($\sigma = 0.115$).	45
3.5	Relative error of the mean task-processing time in the case of a Weibull distribution ($\alpha = 0.548$).	46
3.6	Relative error of the mean task-processing time in the case of a Pareto distribution ($\beta = 2.69$).	46
3.7	Optimal number of replications for the mean task-processing time with respect to the coefficient of variation in the case of a hyper-exponential distribution.	47
3.8	Optimal number of replications for the mean task-processing time with respect to the coefficient of variation in the case of a Weibull distribution.	48
3.9	Optimal number of replications for the mean task-processing time with respect to the coefficient of variation in the case of a Pareto distribution.	48
3.10	Ratio of the mean task-processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a hyper-exponential distribution.	49
3.11	Ratio of the mean task-processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Weibull distribution.	49
3.12	Ratio of the mean task-processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Pareto distribution.	50
3.13	Ratio of the mean total processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a hyper-exponential distribution.	51

3.14	Ratio of the mean total processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Weibull distribution.	51
3.15	Ratio of the mean total processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Pareto distribution.	52
3.16	Mean task-processing time with simulation in the case of a Pareto distribution ($\beta = 2.69$, $M = 3 \times 10^5$).	55
3.17	Mean total processing time with simulation in the case of a Pareto distribution ($\beta = 2.69$, $M = 3 \times 10^5$).	55
4.1	Processing of a subtask with checkpointing method.	61
4.2	Mean task-processing time with respect to the number of checkpoints for various M ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.	69
4.3	Mean task-processing time with respect to the number of checkpoints for various b ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.	69
4.4	Mean task-processing time with respect to the number of checkpoints for various c ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.	70
4.5	Mean task-processing time with respect to the number of checkpoints for various f ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison between the results of analysis and simulation.	70
4.6	Mean task-processing time with respect to the number of checkpoints for various r ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $c = 300$ [sec]): Comparison between the results of analysis and simulation.	71
4.7	Mean task-processing time with respect to M for the optimal number of checkpoints ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation. . .	73
4.8	Mean task-processing time with respect to b for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation. . . .	73
4.9	Mean task-processing time with respect to c for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation. . . .	74

4.10	Mean task-processing time with respect to f for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.	74
4.11	Mean task-processing time with respect to r for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $b = 24$ [hour], $f = 30$ [day]): Comparison between the results of previous and proposal analyses and simulation.	75
4.12	Mean task-processing time with respect to M for the optimal number of checkpoints ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.	76
4.13	Mean task-processing time with respect to b for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.	76
4.14	Mean task-processing time with respect to c for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.	77
4.15	Mean task-processing time with respect to f for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.	77
4.16	Mean task-processing time with respect to r for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $b = 24$ [hour], $f = 30$ [day]): Comparison among three distributions for the time intervals between consecutive worker failures.	78
4.17	Mean task-processing time with respect to small f for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $f = 1$ to 7 [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.	78

List of Tables

2.1	Values of α and β	18
3.1	Parameter set.	42
3.2	Values of σ , α and β	42
3.3	Standard deviation of the task-processing time.	53
4.1	Parameter set.	68

Chapter 1

Introduction

1.1 Overview of Cloud Computing

Cloud computing has become a new paradigm for utilizing computing resources (e.g. processors, storages, networks, etc.) in recent years. According to the literature [19, 37, 53], cloud computing has the following features.

Sharing resource pool Computing resources are pooled by providers and shared with multiple users. Providers can benefit from economies of scale because they usually utilize an enormous amount of resources in their data centers. On the other hand, users can leave the maintenance of resources to providers who are expected to have the better knowledge to manage them.

Rapid elasticity Users can consume computing resources as much as they want. That is, computing resources allocated to users can flexibly scale out and in. In general, providers offer pay-per-use services with a low or no up-front fee, and thus the total amount of expenses for computing operations decreases. Moreover, this elasticity allows users to deal with a surge of service demands easily.

Automatic operation When users employ computing resources, complicated interactions are not necessary and most of operations are automatically conducted. Both users and providers are freed from manual operations, and they can assign human resources to other tasks.

Accessibility via network Computing resources are provided through the network, especially Internet. This enables users to receive services easily on their various devices (e.g. desktops, laptops, tablets, smartphones, etc.).

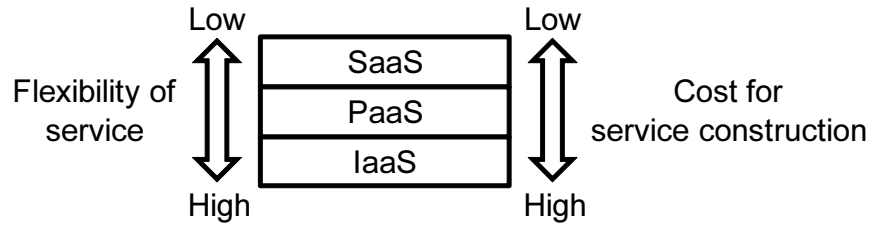


Figure 1.1: Service layers of cloud computing.

Additionally, cloud computing is usually described from two viewpoints: service layers [19, 50] and deployment scopes [19, 53].

Services of cloud computing can be classified into three layers: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) in order from top (see Fig. 1.1). Services which belong to the same layer are in the same abstraction level and target the same user group. Moreover, services of a higher layer can be implemented on those of a lower one. We then illustrate each layer in detail.

SaaS Services in this layer are released as applications to end users through the network. These applications are deployed on providers' servers rather than users' various devices. Therefore, it is easy for providers to test, maintain and upgrade applications. On the other hand, users can utilize applications without the restriction due to their device ability.

PaaS A platform to develop applications is provided and main users of PaaS are providers of SaaS. This platform provides application programming interfaces to save the time and effort of developers, and supports not only the development but also deployment and maintenance of applications.

IaaS In this layer, computing resources are directly served in the form of virtual machines and users can freely construct both platforms and applications. Virtualization is a key technology to provide flexible resources and isolate users from managing real physical resources.

Note here that services of a lower layer are more flexible than those of a higher one, while using services of higher layers can reduce the time and expenses to construct services. Therefore, users need to select services of the appropriate layer according to their budgets and required operations.

Services of cloud computing can also be classified according to their scope of deployment (see Fig. 1.2).

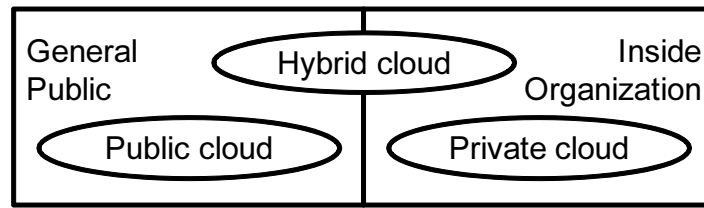


Figure 1.2: Deployment scopes of cloud computing.

Public clouds Providers release their services to the general public. Services can be used with a low or no fee, while users do not have the permission to finely adjust the configurations of these services.

Private clouds Services are exclusively used by a single organization and managed by the organization or an external provider. This enables the organization to freely tune the functions and security settings of services because the organization has the full control of these services. However, this model lacks the benefits of cloud computing, such as a low up-front investment.

Hybrid clouds This model is a combination of public and private clouds that compensates for the drawbacks of each model. For example, operations which require high security or special configurations are conducted on private clouds, and ordinary ones are processed on public clouds.

As described above, the emergence of cloud computing brings great benefits to us. However, there still exist issues which have not been fully addressed [4, 19, 53]. We mention three typical issues below.

Performance evaluation Services of cloud computing are usually deployed on huge computing resources and utilized by a lot of users. This feature causes performance heterogeneity and frequent failures of resources, and makes it difficult to predict and optimize performance of services. Therefore, it is necessary to construct and evaluate models which consider these uncertainties.

Security audit Especially in public clouds, users process applications and store data at providers' resources which are shared with others, and this may cause unexpected security issues. Providers try every possible means to enhance the security of their services. However, in general, it is difficult for users to confirm that their activities are kept confidential, and the function is required which enables to audit security settings.

Interoperability In many cases, providers implement services using their own proprietary technology, and we cannot coordinate services of different providers. This narrows the

choices of services and makes it difficult for users to select appropriate services according to their situations. To address this issue, there are some discussions about standardization and open sourcing of services.

1.2 Overview of Large-Scale Parallel-Distributed Processing

In the last three decades, the amount of generated data has been explosively increasing with the development of technologies, such as sensors, multimedia, scientific experiments and social network services. This significant increase makes it difficult to capture, store and process data by using traditional approaches within a tolerable scope (e.g. time, budget, capacity, etc.). On the other hand, these enormous and various data include precious information which gives us many opportunities to make great benefits, and their analysis results are utilized for forecasting, decision making and process optimization. These massive data which conceal great values are referred to as big data, and effective use of them is an important issue [7, 8].

In order to deal with extreme-scale data, scaling out is a more reasonable and realistic option than scaling up because a more powerful server is highly expensive or cannot be produced due to physical limitations. Therefore, various kinds of parallel-distributed processing schemes are implemented for data processing, utilizing the features of cloud computing, such as resource pooling, elasticity and automation. Implementations of these parallel-distributed processing can be classified into three types [7].

Batch type This type is used for general purposes. Data to be processed are stored in advance and the procedure of processing them is relatively configurable. Implementations of batch type can process the enormous amount of data per unit time, while the time to prepare the processing causes a high latency.

Stream type Some applications need to collect data periodically and process them in real-time. For example, to detect system errors promptly, measurement data should be continuously monitored. In order to meet the requirements of such applications, stream-type implementations can process data which arrive constantly in very short response time. However, this type cannot use past records because most of arriving data are discarded after the processing.

Interactive type Some users need to analyze data interactively. In this case, data including past records should be processed in short response time according to the demands of users. Implementations of interactive type provide functions which meet such require-

ments, limiting the procedure of processing data to the extent which is necessary for analysis.

In the following, we mainly focus on batch type and briefly describe an actual implementation. MapReduce [14] is one of the most successful implementations for batch-type parallel-distributed processing. In this implementation, there are two kinds of nodes: a master and workers. The master is responsible for scheduling and monitoring tasks of data processing which are requested by users. A task is split into a number of subtasks, and they are assigned to and executed by workers. The procedure of MapReduce is mainly composed of two steps: Map and Reduce, which are inspired by the functions commonly used in functional languages. Data are treated in the form of key/value pairs.

In Map step, the master assigns Map subtasks to workers. These workers read input data which are allocated by the master, and apply the user-defined function to each key/value pair. The results are then locally sorted by key and stored as intermediate data. Reduce step begins after all Map subtasks are completed. Reduce subtasks are also assigned to workers by the master, and these workers read and aggregate intermediate data of the same key in the user-defined way. In this procedure, each worker is responsible for a different set of keys. Finally, the results of this step are generated as output data. We illustrate the execution overview of MapReduce in Fig. 1.3.

MapReduce has the following advantages in comparison with traditional implementations [34, 42].

Fault tolerance MapReduce assumes that the processing is executed on huge clusters of commodity servers. In such an environment, failures of servers occur frequently. Therefore, in MapReduce, the master schedules and monitors the processing and workers report their status. If there are worker failures, the master automatically re-assigns failed subtasks to other workers.

Scalability Map (resp. Reduce) subtasks are executed by workers without data exchange between other Map (resp. Reduce) subtasks. That is, each subtask can be independently processed in parallel. Moreover, MapReduce is implemented on a loosely-coupled system, and it is easy to add and remove servers to the system. These features achieve a high scalability.

User friendliness Users are required only to program functions which are applied on Map and Reduce steps, and do not need to worry about the division of a task and the assignment of subtasks. The parallelization of the processing is automatically conducted by the master.

On the other hand, there are challenges to overcome on batch-type parallel-distributed processing such as MapReduce [32, 34].

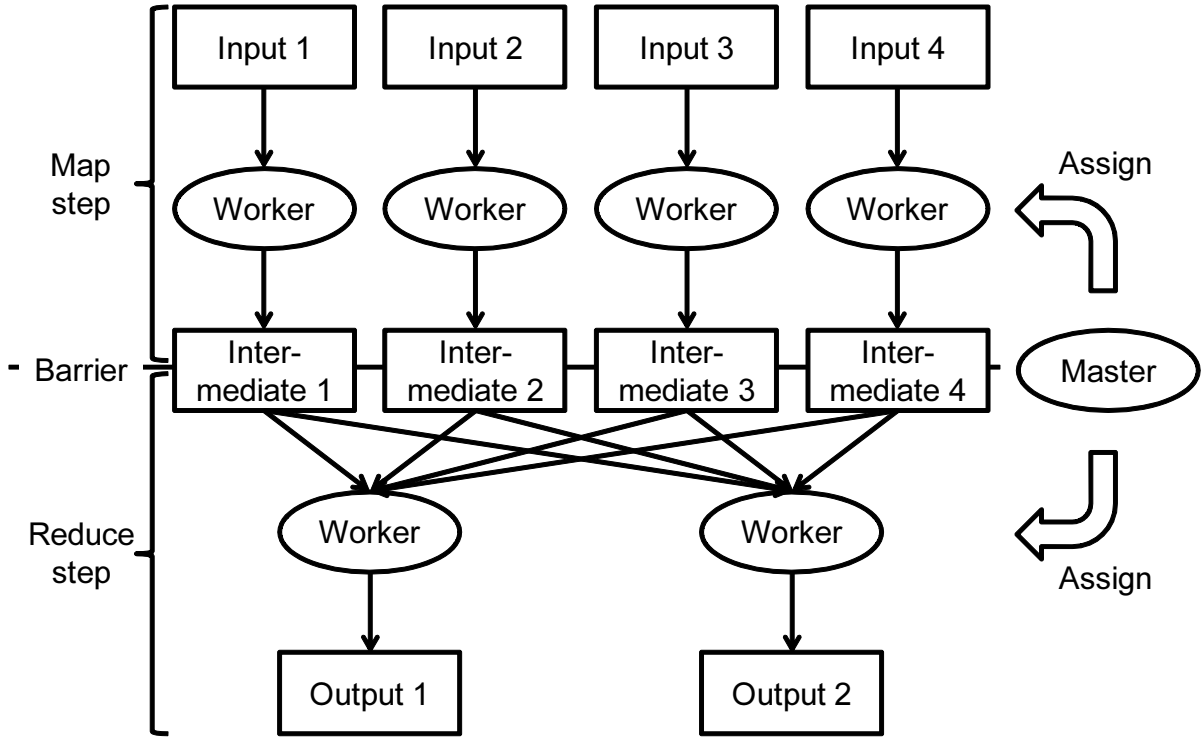


Figure 1.3: Execution overview of MapReduce.

Latency Implementations of batch type mainly aim at high scalability and throughput, and do not pay much attention to the latency. Therefore, it is pointed out that there is room for improvement on the procedures of task preparation, scheduling and monitoring which cause a high latency.

Efficiency Batch-type parallel-distributed processing can treat the high volume of data per unit time, while the amount of processing data per the number of workers is relatively low. This is partly because fault tolerance schemes cause a high overhead.

Parameter tuning This issue is for all types of parallel-distributed processing. There are a number of configuration parameters which affect the performance of the processing. The setting of these parameters usually relies on a rule of thumb which is not fully optimized, and the function of automatic tuning is required.

1.3 Outline of the Dissertation

In the previous two sections, we summarize the features of parallel-distributed processing on cloud computing and illustrate the issues to be addressed. In this dissertation, we especially focus on the issue of stragglers and its improvement methods, which are described below.

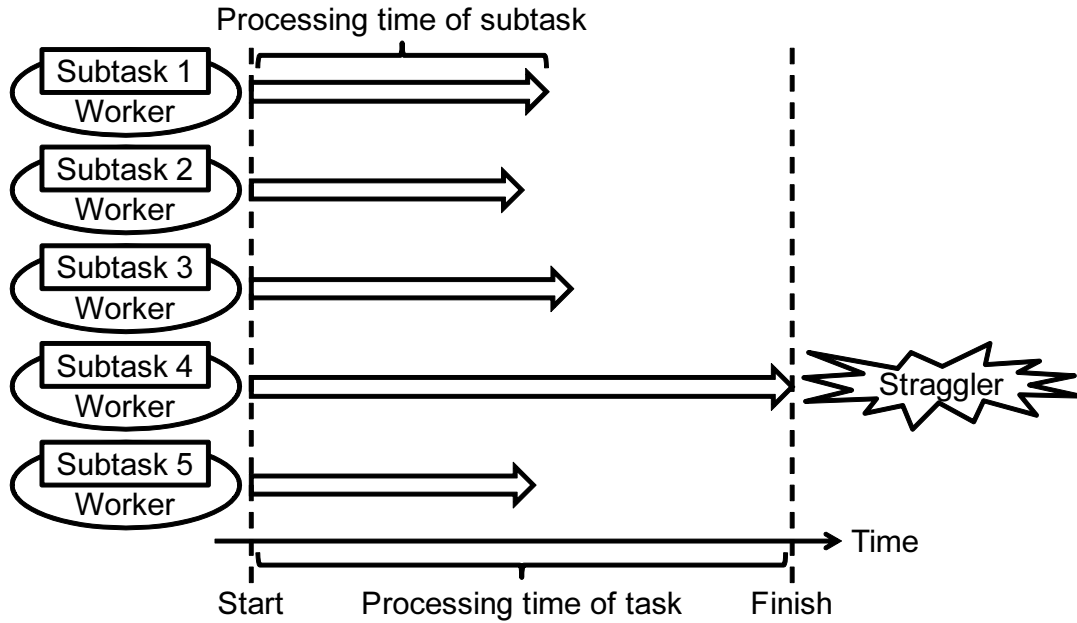


Figure 1.4: Overview of the issue of stragglers.

In parallel-distributed processing, a task is split into a number of subtasks and those are processed independently in parallel on a cluster of machines referred to as workers. The task completes when all the subtasks have finished, and thus subtasks which take longer for the execution result in delay in the processing of the task (see Fig. 1.4). This performance degradation is referred to as the issue of stragglers [14], and becomes more serious as the number of subtasks increases.

There are two well-known methods to alleviate the issue of stragglers: task replication [11] and checkpointing [36]. Workers usually suffer the unexpected and temporal degradation of the processing performance in cloud computing because a lot of users share huge computing resources. The purpose of task replication is to decrease the processing time of subtasks involved in such a performance degradation of workers. In task replication, each subtask is replicated, and the original subtask and its copies are assigned to multiple workers separately. The processing of the subtask ends when either the original subtask or its copy is completed (see Fig. 1.5). Task replication mitigates the impact of ill-conditioned workers and decreases the processing time of subtasks.

On the other hand, checkpointing aims to deal with the increase in the processing time of subtasks caused by worker failures, which occur frequently in cloud computing because of its huge system scale. In general, when a worker fails, the subtask involved in its failure is re-executed by another worker from the beginning. In contrast, in checkpointing, the progress of the processing is periodically saved as a checkpoint during a period of the processing of a

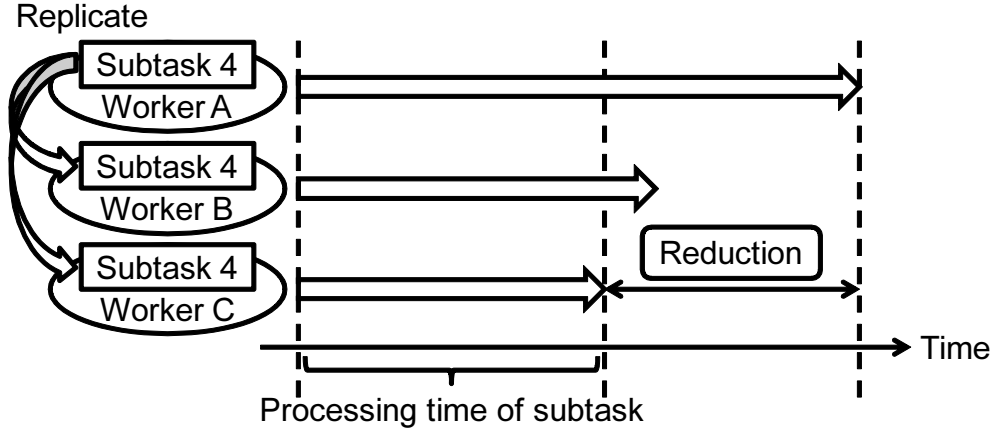


Figure 1.5: Overview of task replication.

subtask. When worker failure occurs, the subtask processed by the failed worker is resumed by another worker from the latest checkpoint (see Fig. 1.6). Checkpointing prevents the failed subtask executing from the beginning of the processing and reduces the increase in the processing time of subtasks.

In this dissertation, we consider the impact of stragglers and the effect of task replication and checkpointing on the performance of parallel-distributed processing through analytical models and their evaluations. In order to evaluate these performances, there are also approaches based on simulation or measurement. However, when a system is extreme-scale, these approaches require excessive time or computing resources to evaluate the performance, and sometimes lead the system to local optimum configurations. Therefore, through analytical studies, we aim to evaluate the performance without using a lot of resources, and give directions for further optimization of system configurations.

The rest of the dissertation is organized as follows.

In Chapter 2, we consider the impact of stragglers and the efficiency of backup-task scheduling, which is a kind of task-replication scheme. We model a task-scheduling server as a single-server queue, in which the server consists of a number of workers. A task entering the service facility is split into subtasks, and the task service ends when all the subtasks are completed. We then consider two task-scheduling policies: normal scheduling and backup-task one. For normal scheduling, each subtask is served by its own worker. In backup-task scheduling, on the other hand, each subtask is processed not only by its own worker but also by an alternative distinct worker, and the subtask service ends when either of the two workers' processes is completed. In both scheduling policies, we explicitly derive distributions of the processing time of a task when the processing time of a subtask by a worker follows a Weibull or Pareto distribution. We then derive the maximum throughput, mean response time

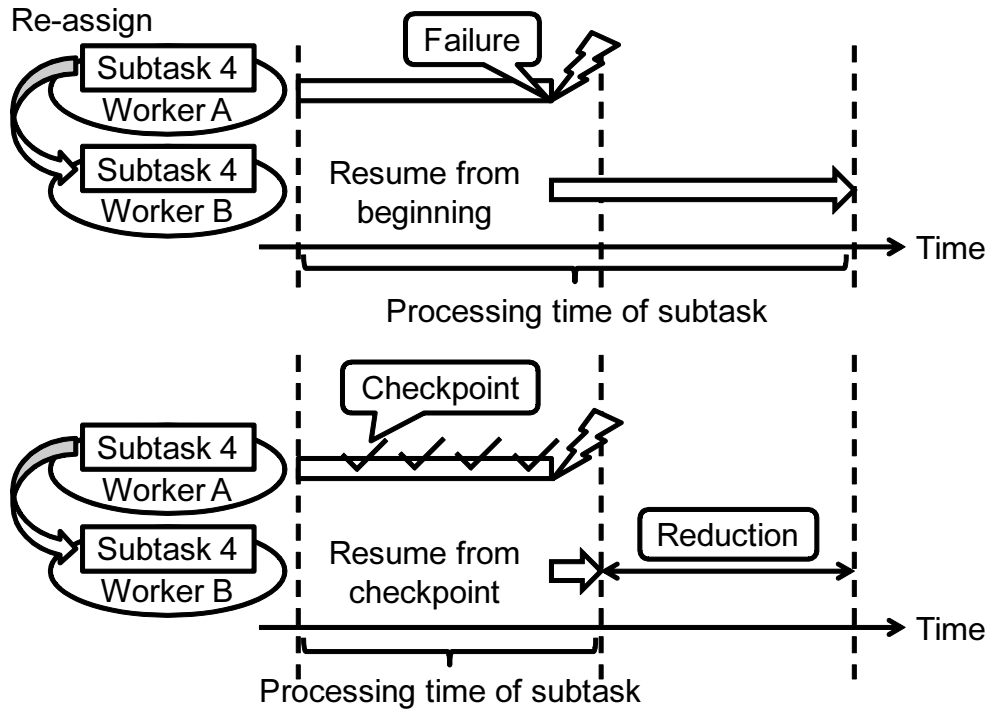


Figure 1.6: Overview of checkpointing.

and mean total processing time. In numerical examples, we investigate the impact of stragglers and discuss the efficiency of backup-task scheduling. This study is mainly based on [26].

Chapter 3 evaluates the effect of task-replication scheduling in terms of the number of replications. We consider the task-replication scheduling policy in which a task entering the service facility is split into subtasks, and the task service ends when all of the subtasks are completed. Moreover, each subtask is processed not only by its own worker but also by alternative distinct workers, and the subtask service ends when one of the relevant workers' processes is completed. For this system, we propose an approach based on extreme value theory for approximately deriving the mean value and standard deviation of the processing time of a task. Moreover, we exactly derive the mean total processing time. We then consider cases in which the processing time of a subtask by a worker follows a hyper-exponential, Weibull or Pareto distribution. In numerical examples, we discuss the effect of task replication on the performance measures in terms of the number of replications. This study is mainly based on [27].

In Chapter 4, we evaluate the effect of checkpointing method on the processing time of a task, and consider the optimal number of checkpoints which achieves the shortest processing time of a task. We construct a stochastic model for parallel-distributed processing with checkpointing, in which a task accepted by the service facility is split into subtasks, and the

processing of the task ends when all of the subtasks are completed. Moreover, each subtask's checkpoints are made periodically during its processing, and a subtask is resumed by another worker from the latest checkpoint when the dedicated worker fails. For this system, we approximately derive explicit expressions for the mean processing time of a task and the optimal number of checkpoints. In numerical examples, we show the usefulness of the derived optimal number of checkpoints. This study is mainly based on [28].

Finally, Chapter 5 concludes the dissertation and suggests some directions for future research.

Chapter 2

Performance Degradation by the Issue of Stragglers

2.1 Introduction

Recently, cloud computing has attracted considerable attention due to the emergence of huge computing resources and its significant improvement in usage fee. In [4], cloud computing is defined as the sum of the existing concepts, software as a service (SaaS) and utility computing. More precisely, cloud computing is the combined concept of providing a computer-processing service only as needed via the Internet (SaaS) and using server resources in a data center only as needed (utility computing). A remarkable feature of cloud computing is that data centers providing cloud computing services have a huge number of computing resources, and this number is still increasing. For example, Google aims to have several million machines in their data centers [15].

In addition, volunteer computing is becoming popular due to the spread of computing resources with an Internet connection. In volunteer computing, distributed computing resources are donated by individuals as well as organizations, and, for some projects, the number of hosts is in the hundreds of thousands [29]. Therefore, in both cloud and volunteer computing, efficient use of an extremely large number of computing resources is a critical issue.

With the increase in the capacity of hard-disks, computing tasks must handle a greater volume of data, and an enormous amount of time is required if a task is carried out by an individual computing resource. In cloud and volunteer computing, an enormous amount of data is handled by a huge number of computing resources in parallel-distributed processing fashion [5, 14, 43]. This scheme is used for data mining, document processing and machine learning, and is used by numerous companies and organizations for processing large-scale data [42]. In the following, we refer to this processing mechanism as *large-scale parallel-*

distributed processing.

In large-scale parallel-distributed processing, a huge task is split into a number of subtasks and those are processed independently in parallel on a cluster of machines referred to as workers. The huge task completes when all the subtasks have finished. Therefore, workers that take longer to process their assigned subtasks result in delay in the processing of the task (the issue of stragglers) [14]. One of the reasons causing slow workers is frequent machine failure because data centers consist of a huge number of commodity machines for reducing hardware cost [5, 15]. Moreover, it is reported in [51] that virtualization technology can cause resource competition, and as a result, heterogeneity occurs in the processing speed of workers. In the following, we refer to the time to complete a task (resp. subtask) as the *task-processing* (resp. *subtask-processing*) time.

In order to alleviate the issue of stragglers, there exist two scheduling schemes: load balancing [20] and backup tasks [14]. In load balancing, the subtask size for a worker is determined according to its processing speed. In other words, small subtasks are allocated to slow workers, while large subtasks are performed by fast workers. This scheduling makes the variance of the subtask-processing times significantly small. However, the load-balancing scheduler must know each worker's subtask-processing time a priori.

In backup-task scheduling, on the other hand, backup executions of the remaining in-progress subtask are conducted when the elapsed time of the processing of the subtask is greater than a pre-specified threshold. The processing of the subtask ends when either the original subtask or backup execution is completed. One advantage of this scheduling is that the backup-task scheduler activates backup executions for a worker according to the elapsed time of the processing of a subtask, i.e., no a priori information about the subtask-processing time is needed.

There is much literature on cloud computing, and most of studies are concerned with service platform and cost efficiency from the economical point of view. There are a few studies for performance issues on cloud computing, and those are based on measurement-based analysis. (See, for example, [17, 48].) In terms of the theoretical approach to performance issues on cloud computing, Xiong et al. [47] consider a queueing network model which consists of a Web-server queue and a service-center queue. Focusing on the percentile of the response time as a performance measure of cloud computing, they approximately analyze the response time distribution. In their model, however, the service-center part is modeled as a single-server queue with a fixed service rate, and this model is too simple to describe large-scale parallel-distributed processing.

In [20], Dobber et al. investigate the effectiveness of dynamic load balancing (DLB) and job replication (JR) by trace-driven simulation experiments, proposing a hybrid scheduling scheme of DLB and JR. Cirne et al. [11] also investigate the effectiveness of several

job-replication schedulers by simulation, comparing those with traditional information-based schedulers. Note that most of related work is concerned with the performance of task schedulers by simulation. To the best of the author's knowledge, the effect of backup-task scheduling on the improvement of the task-processing time has not been fully studied yet.

In this chapter, we consider the impact of stragglers and the efficiency of backup-task scheduling. We focus on a task-scheduling server in which tasks are processed in first-come, first-served (FCFS) order. We model the task-scheduling server as a single-server queue, in which the server consists of a number of workers. A task entering the service facility is split into subtasks of an equal size. Note that the assumption of equally sized subtasks becomes reasonable when huge sized input data are split into data pieces of approximately equal size [14]. The task service ends when all the subtasks are completed.

We consider two task-scheduling policies: normal scheduling and backup-task one. For normal scheduling, each subtask is served by its own worker. In backup-task scheduling, on the other hand, each subtask is processed not only by its own worker but also by an alternative distinct worker, and the subtask service ends when either of the two workers' processes is completed. In the following, we assume that the times to complete subtask by its own worker and an alternative distinct worker are independent and identically distributed (i.i.d.), and refer to these times as the *worker-processing times*. In both scheduling policies, we explicitly derive distributions of the task-processing time when the worker-processing time follows a Weibull or Pareto distribution. We use these distributions according to the measurement of real systems [2, 9, 10]. We then derive the maximum throughput, mean response time, and mean total processing time. In numerical examples, we validate the analysis by Monte Carlo simulation. We then investigate the impact of stragglers under normal scheduling. Moreover, we compare the performance measures under backup-task scheduling with that under normal scheduling, discussing the efficiency of backup-task scheduling.

This chapter is organized as follows. In Section 2.2, analytical models for two scheduling policies are described. In Section 2.3, we derive performance measures. Section 2.4 shows numerical examples of derived performance measures. Finally, we summarize this chapter in Section 2.5.

2.2 Analytical Models for Two Scheduling Policies

We consider two analytical models for large-scale parallel-distributed processing: *normal processing model* and *backup-task processing model* (referred to as Models N and B, respectively, hereafter). In each model, the system consists of an infinite buffer and a server with workers. Tasks arrive at the system according to a Poisson process with rate λ , and they are processed

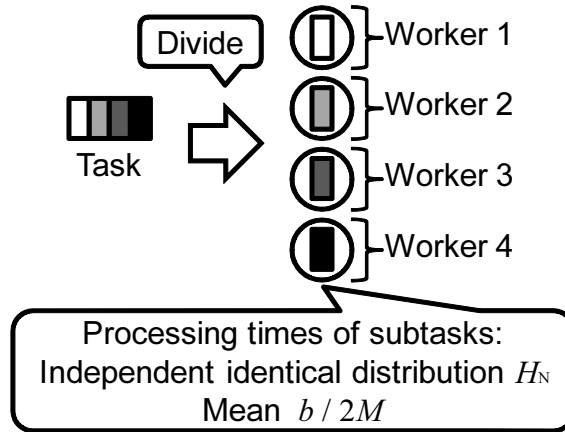


Figure 2.1: Normal processing model (Model N).

on the FCFS basis.

The details of the two models are as follows:

(i) Normal processing model (Model N)

The server has $2M$ (which is a positive integer) workers, and a task is divided into $2M$ subtasks. Each subtask is processed by a worker, and its processing time follows a distribution function H_N with mean $b/(2M)$ ($b > 0$), independently of those of the other subtasks (Fig. 2.1). Further, the processing time of a task (consisting of $2M$ subtasks) is defined as the maximum of the processing times of the $2M$ subtasks generated from the task. As a result, the task-processing times are i.i.d. with a distribution function G_N , which is given by

$$G_N(t) = \{H_N(t)\}^{2M}, \quad t \geq 0.$$

(ii) Backup-task processing model (Model B)

The server consists of M pairs of workers. A task is divided into M subtasks, from each of which a backup subtask is replicated. An original subtask and its backup subtask are assigned to a pair of workers separately. The processing times of the $2M$ subtasks (including M backup subtasks) generated from a task are i.i.d. with a distribution function H_B with mean b/M (Fig. 2.2). The processing of each pair of an original subtask and its backup subtask is finished when either of them is completed. The task-processing time is defined in the same way as Model N. Thus, the task-processing times are i.i.d. with a distribution function G_B , which is given by

$$G_B(t) = [1 - \{1 - H_B(t)\}^2]^M, \quad t \geq 0.$$

In what follows, we refer to H_N and H_B as *worker-processing time distributions*.

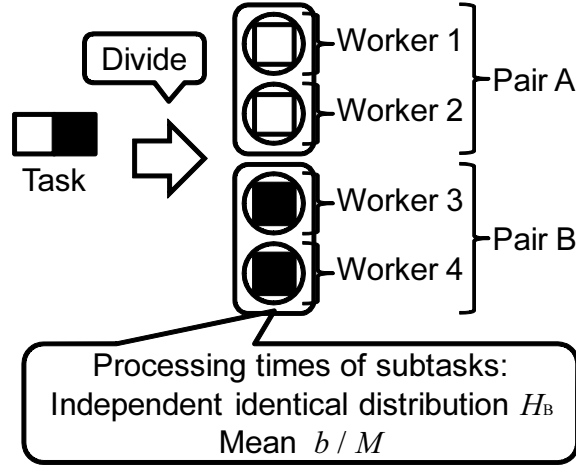


Figure 2.2: Backup-task processing model (Model B).

2.3 Analysis

2.3.1 Performance Measures

We consider three performance measures: the maximum throughput, mean response time, and mean total processing time. The maximum throughput is defined as the reciprocal of the mean value of the task-processing time; the mean response time as the mean sojourn time of a task in the system (from its arrival to its departure); and the mean total processing time as the mean of the total running time of the $2M$ workers during the processing of a task.

Let subscript “x” denote the index symbol for the two processing models described in the previous section, i.e., $x = N$ or B . Let T_x , W_x and P_x ($x = N, B$) denote the maximum throughput, mean response time and mean total processing time, respectively, in Model x . Note here that Model x is considered as an FCFS $M/G/1$ queue (Fig. 2.3), where the service time distribution is given by G_x . We then have (see, e.g., [24])

$$T_x = \frac{1}{g_x^{(1)}}, \quad W_x = \frac{\lambda g_x^{(2)}}{2(1 - \lambda g_x^{(1)})} + g_x^{(1)}, \quad x = N, B, \quad (2.1)$$

where $g_x^{(1)}$ and $g_x^{(2)}$ denote the first and second moments of a distribution function G_x , i.e.,

$$g_x^{(1)} = \int_0^\infty t dG_x(t), \quad g_x^{(2)} = \int_0^\infty t^2 dG_x(t),$$

respectively. From the definition, we also have

$$P_x = \begin{cases} E \left[\sum_{i=1}^{2M} U_{x,i} \right] = 2ME[U_{x,1}], & x = N, \\ E \left[\sum_{i=1}^M 2 \min \{U_{x,2i-1}, U_{x,2i}\} \right] = 2ME[\min \{U_{x,1}, U_{x,2}\}], & x = B, \end{cases}$$

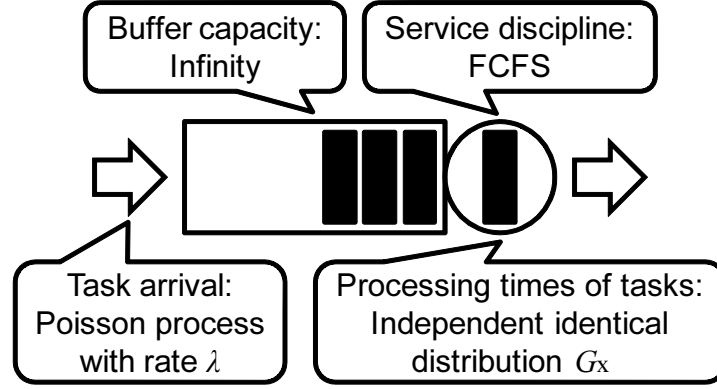


Figure 2.3: Analytical model for large-scale parallel-distributed processing.

where $U_{x,i}$ ($i = 1, 2, \dots, 2M$) denotes the worker-processing time of i th worker. Note that the $U_{x,i}$'s are i.i.d. random variables with a distribution function H_x .

2.3.2 Special Cases for the Worker-Processing Time Distribution

We consider two types of the worker-processing time distribution. For convenience, let $M_N = 2M$ and $M_B = M$.

(a) Weibull distribution

$$H_x(t) = 1 - \exp\{-(t/\eta_x)^\alpha\}, \quad t \geq 0,$$

with scale parameter $\eta_x = b/\{M_x \Gamma(1 + 1/\alpha)\}$ ($\alpha > 0$); and

(b) Pareto distribution

$$H_x(t) = \begin{cases} 1 - (\mu_x/t)^\beta, & t \geq \mu_x, \\ 0, & 0 \leq t < \mu_x, \end{cases}$$

where $\mu_x = b(\beta - 1)/(M_x \beta)$ ($\beta > 2$).

For the two worker-processing time distributions, we can calculate the moments $g_N^{(n)}$ and $g_B^{(n)}$ ($n = 1, 2$) as follows:

(a) Weibull distribution

(i) Model N

$$g_N^{(1)} = \frac{b}{2M} \sum_{k=1}^{2M} (-1)^{k-1} \binom{2M}{k} \frac{1}{k^{\frac{1}{\alpha}}},$$

$$g_N^{(2)} = \frac{\Gamma(1 + \frac{2}{\alpha})}{\{\Gamma(1 + \frac{1}{\alpha})\}^2} \frac{b^2}{4M^2} \sum_{k=1}^{2M} (-1)^{k-1} \binom{2M}{k} \frac{1}{k^{\frac{2}{\alpha}}};$$

(ii) Model B

$$g_B^{(1)} = \frac{b}{2^{\frac{1}{\alpha}} M} \sum_{k=1}^M (-1)^{k-1} \binom{M}{k} \frac{1}{k^{\frac{1}{\alpha}}},$$

$$g_B^{(2)} = \frac{\Gamma(1 + \frac{2}{\alpha})}{\{\Gamma(1 + \frac{1}{\alpha})\}^2} \frac{b^2}{2^{\frac{2}{\alpha}} M^2} \sum_{k=1}^M (-1)^{k-1} \binom{M}{k} \frac{1}{k^{\frac{2}{\alpha}}};$$

(b) Pareto distribution

(i) Model N

$$g_N^{(1)} = (\beta - 1) \frac{b}{2M} \sum_{k=1}^{2M} (-1)^{k-1} \binom{2M}{k} \frac{k}{\beta k - 1},$$

$$g_N^{(2)} = \frac{(\beta - 1)^2}{\beta} \frac{b^2}{4M^2} \sum_{k=1}^{2M} (-1)^{k-1} \binom{2M}{k} \frac{k}{\beta k - 2};$$

(ii) Model B

$$g_B^{(1)} = 2(\beta - 1) \frac{b}{M} \sum_{k=1}^M (-1)^{k-1} \binom{M}{k} \frac{k}{2\beta k - 1},$$

$$g_B^{(2)} = \frac{2(\beta - 1)^2}{\beta} \frac{b^2}{M^2} \sum_{k=1}^M (-1)^{k-1} \binom{M}{k} \frac{k}{2\beta k - 2}.$$

Combining (2.1) with the above equations, we can obtain T_x and W_x . We can also calculate P_x as follows:

(a) Weibull distribution

$$P_x = \begin{cases} b, & x = N, \\ 2^{(1-\frac{1}{\alpha})} b, & x = B; \end{cases}$$

(b) Pareto distribution

$$P_x = \begin{cases} b, & x = N, \\ \frac{4(\beta - 1)}{2\beta - 1} b, & x = B. \end{cases}$$

2.4 Numerical Examples

In this section, we show some numerical examples. First, we discuss the model validity by comparing the analytical results and Monte Carlo simulation. We then investigate the impact of stragglers under normal scheduling. Moreover, we consider the effect of backup-task

Table 2.1: Values of α and β .

Coefficient of variation	8.307	2.236	1.000	0.5227	0.2805
α	0.2500	0.5000	1.000	2.000	4.000
β	2.007	2.095	2.414	3.159	4.702

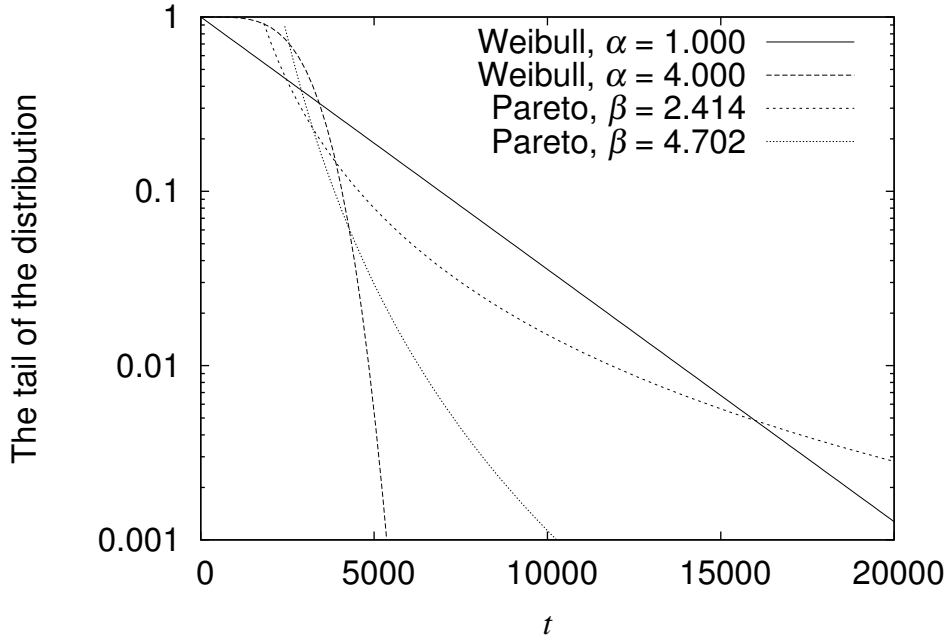


Figure 2.4: Tail of the worker-processing time distribution.

scheduling for improving the issue of stragglers by comparing Models N and B (i.e., the normal processing model and backup-task processing one).

In the following numerical examples, we set $b = 3.000 \times 10^7$ (sec) (i.e. about a year) and $\lambda = 3.000 \times 10^{-8}$ (task/sec). Furthermore, M is varied from 1 to 10,000, and the values of α and β are determined such that the coefficient of variation of the worker-processing time distribution takes the values as shown in Tab. 2.1. Note that the coefficient of variation for a Weibull (resp. Pareto) distribution becomes large with the decrease in α (resp. β), and the tail of a Pareto distribution is heavier than that of a Weibull distribution, although the coefficients of variation are the same (See Fig. 2.4).

2.4.1 Model Validation

In this subsection, we discuss the model validation. In our model of backup-task scheduling (Model B), we assume that alternative subtasks are simultaneously executed when a task

process starts. In real environment, on the other hand, backup executions are activated when the elapsed time of the processing of a subtask is greater than a pre-specified threshold. In order to validate our analytical model, we conducted Monte Carlo simulation experiments. In our simulation setting, a backup execution of a subtask starts when the elapsed time of the processing of the subtask exceeds $\xi b/M$. Here, ξ is set to 1.0, 1.5 and 2.0. We calculated the 95% confidence interval of the maximum throughput, mean response time, and mean total processing time.

Figure 2.5 represents the mean response time in the case of a Pareto distribution with $\beta = 2.007$ against the number of workers in log-log plot. In Fig. 2.5, the mean response time for Model B is smaller than that for simulation, and the difference between them decreases with the increase in the number of workers. This implies that our model gives a lower bound of backup-task scheduling on the mean response time, and the model assumption is valid when the number of workers is large. This trend can be seen for other parameters of Weibull and Pareto distributions. We also confirm the same trend on the maximum throughput.

Figure 2.6 (resp. Figure 2.7) illustrates the mean total processing time for Model B and simulation in the case of a Pareto distribution with $\beta = 2.007$ (resp. Weibull distribution with $\alpha = 0.2500$). The horizontal axis represents the number of workers in log scale. In both figures, we observe that the mean total processing time remains almost constant when the number of workers increases. This is because the size of a task is a constant b and independent of the number of subtasks. Note that the overhead of parallel-distributed processing is not taken into consideration in both analysis and simulation, and the mean total processing time is almost insensitive to the number of workers. In Fig. 2.6, the mean total processing time for Model B is larger than that for simulation, and the difference grows when the backup execution threshold increases. This result suggests that our model gives the worst case on resource consumption. This tendency can be seen for other parameters of a Pareto distribution and a Weibull distribution with $\alpha \geq 1$.

In Fig. 2.7, on the other hand, the mean total processing time for Model B is smaller than that for simulation, and the difference between them decreases with the decrease in the backup execution threshold. This implies that resource consumption is the smallest when backup executions are activated from the beginning of the processing of a task. This trend is the same for a Weibull distribution with $\alpha < 1$.

These results indicate that the maximum throughput and mean response time can be predicted quantitatively with our model when the number of workers is large. On the other hand, our model is not suitable for quantitative evaluation of the mean total processing time. However, the qualitative trend of the mean total processing time can be described well by this model.

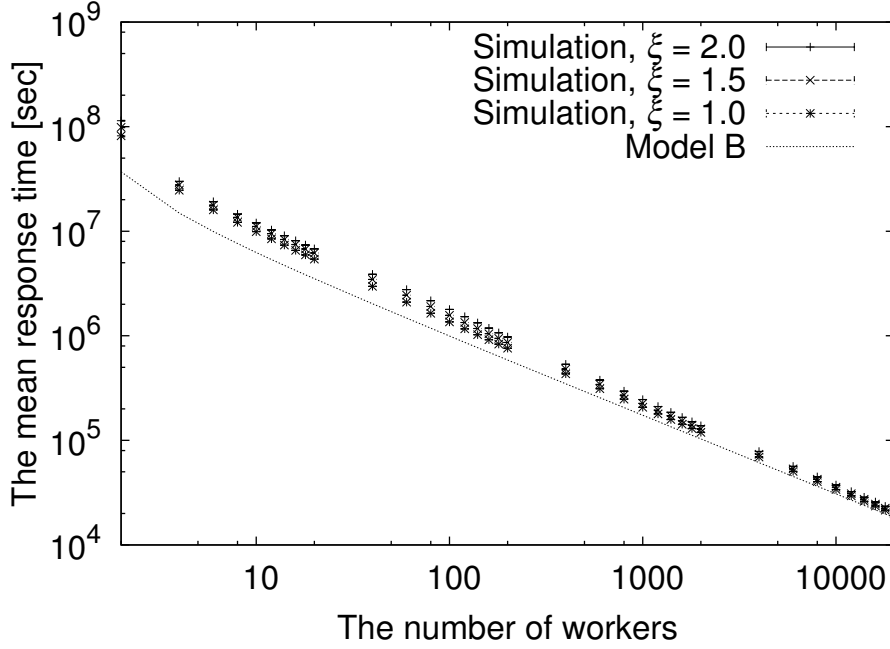


Figure 2.5: Mean response time for Model B and simulation in the case of a Pareto distribution ($\beta = 2.007$).

2.4.2 Impact of Stragglers

In this subsection, we investigate the issue of stragglers by the normal processing model (Model N).

Figures 2.8 and 2.9 represent the maximum throughput for Model N against the number of workers in log-log plot. Here, the worker-processing time distribution is set to a Weibull (resp. Pareto) distribution in Fig. 2.8 (resp. Fig. 2.9). It is observed in Fig. 2.8 (resp. Fig. 2.9) that when α (resp. β) is small, the maximum throughput is less likely to grow with the increase in the number of workers. This is because slow workers are more likely to exist with the increase in the coefficient of variation, and the response time of a task is not significantly improved.

Figures 2.10 and 2.11 illustrate the mean response time for Model N against the number of workers in log-log plot. In both figures, the mean response time is calculated for five values of distribution parameters. Figure 2.10 is the case of a Weibull distribution, while Fig. 2.11 is the case of a Pareto distribution. In both figures, we observe that the mean response time for a large coefficient of variation is less likely to decrease with the increase in the number of workers. This reason is the same as that of the maximum throughput.

These results suggest that the coefficient of variation of the worker-processing time distribution significantly affects the response time performance of large-scale parallel-distributed

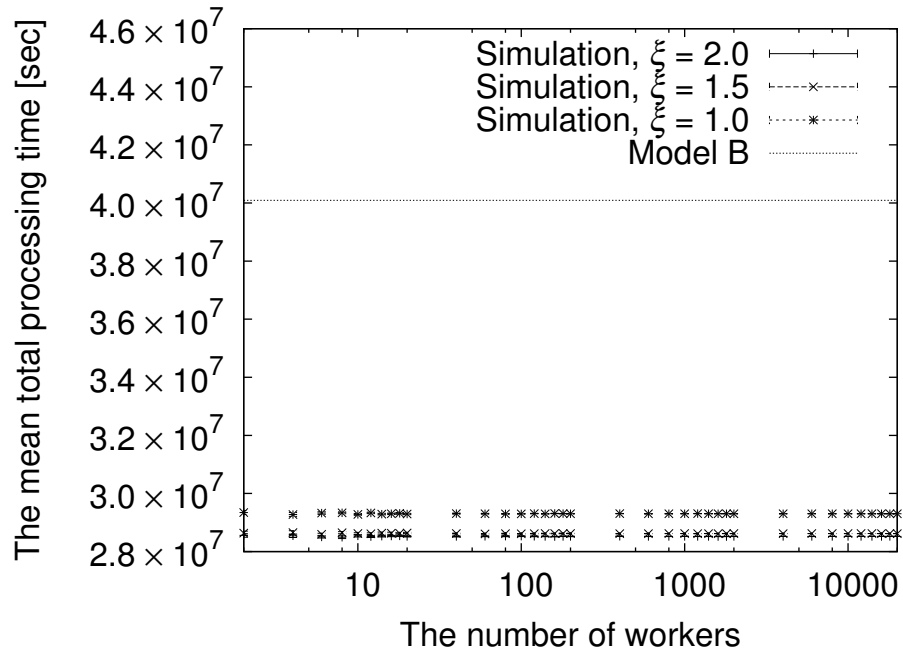


Figure 2.6: Mean total processing time for Model B and simulation in the case of a Pareto distribution ($\beta = 2.007$).

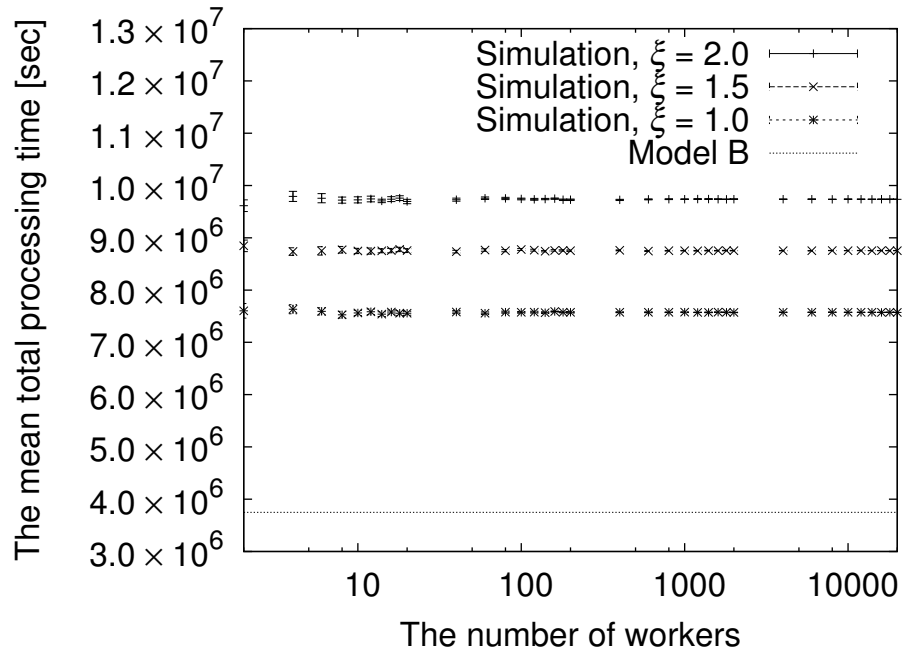


Figure 2.7: Mean total processing time for Model B and simulation in the case of a Weibull distribution ($\alpha = 0.2500$).

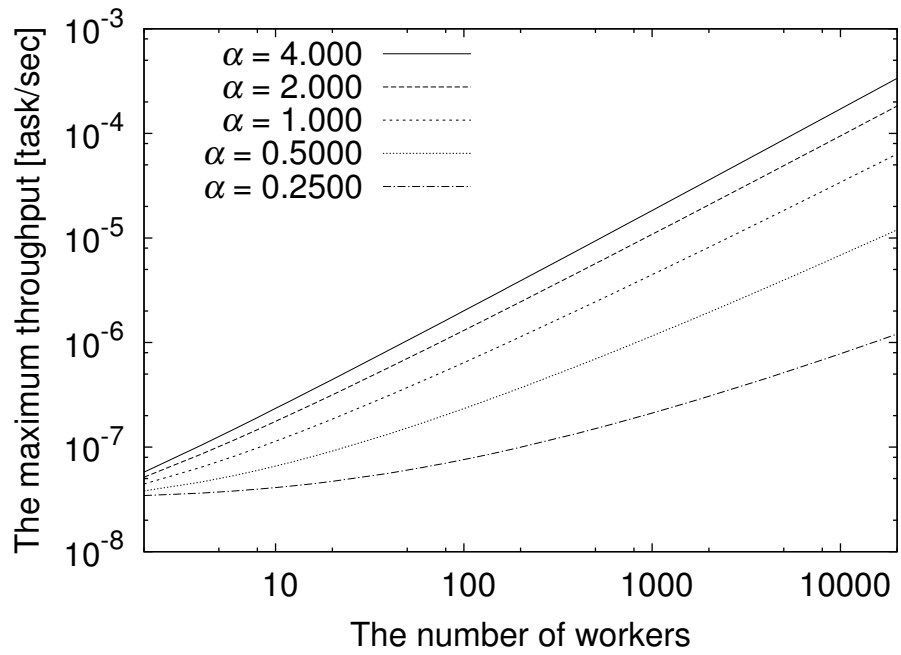


Figure 2.8: Maximum throughput for Model N in the case of a Weibull distribution.

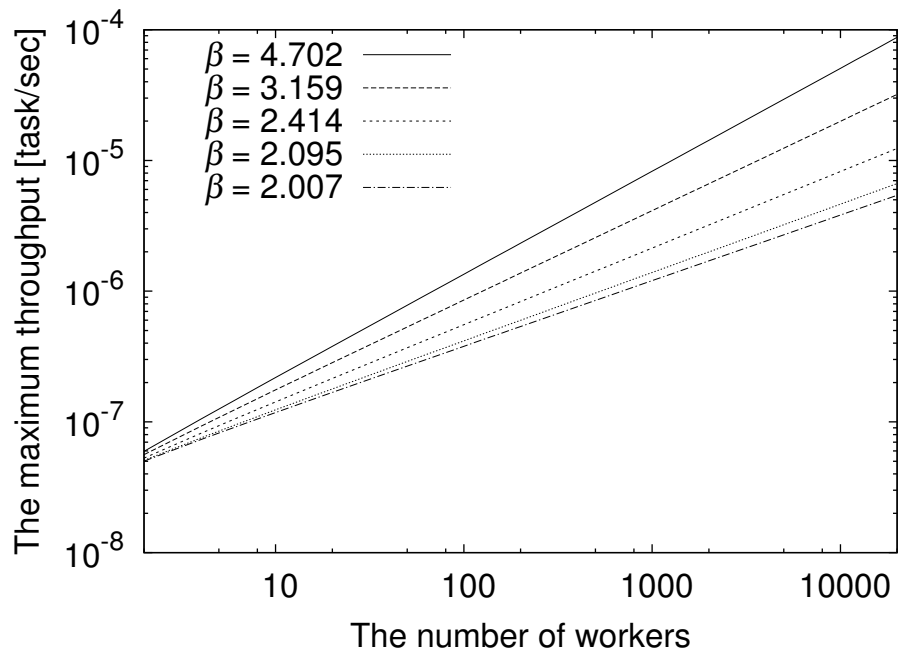


Figure 2.9: Maximum throughput for Model N in the case of a Pareto distribution.

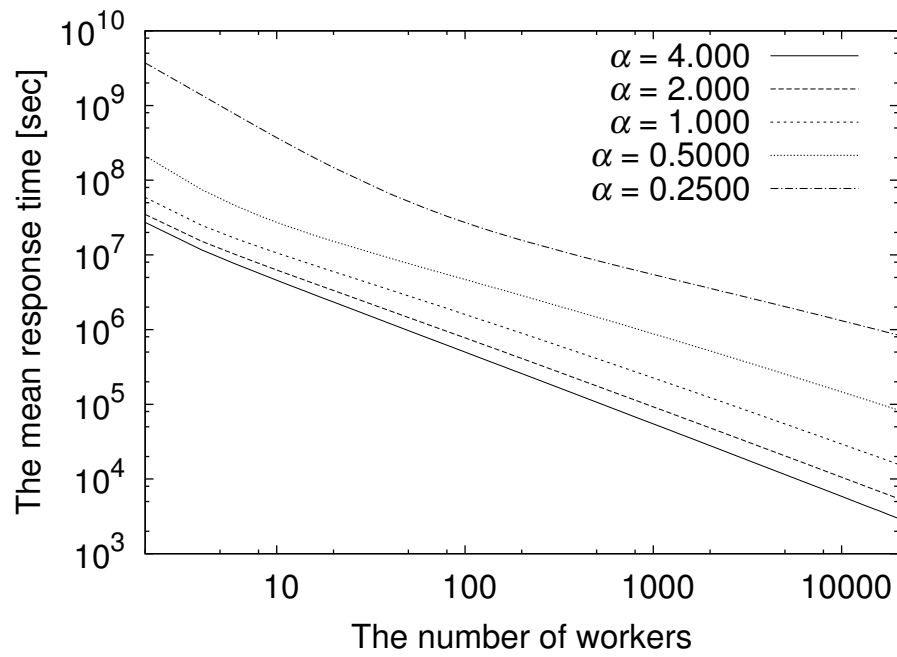


Figure 2.10: Mean response time for Model N in the case of a Weibull distribution.

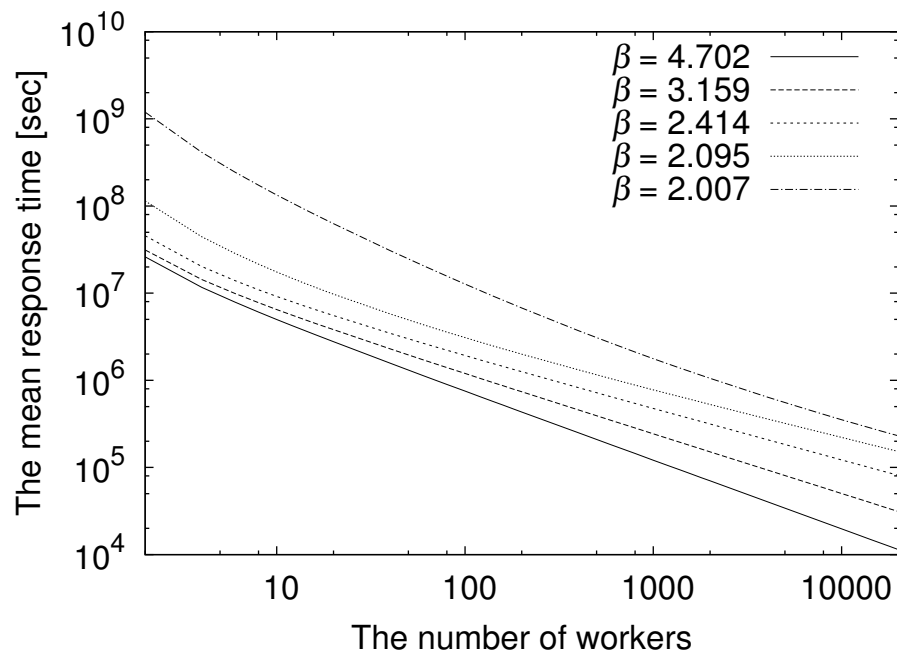


Figure 2.11: Mean response time for Model N in the case of a Pareto distribution.

processing.

2.4.3 Efficiency of Backup-Task Scheduling

In this subsection, we investigate the effect of backup-task scheduling on the performance of large-scale parallel-distributed processing.

Figure 2.12 (resp. Figure 2.13) represents the ratio of the maximum throughput for Model B to that for Model N in the case of a Weibull (resp. Pareto) distribution. The horizontal axis represents the number of workers in log scale, and the ratio in Fig. 2.12 (resp. Fig. 2.13) is calculated in five cases of α (resp. β). In Fig. 2.12, the ratio gradually decreases with the increase in the number of workers. This implies that under backup-task scheduling, increasing the number of workers does not improve the throughput performance effectively when the worker-processing time follows a Weibull distribution. We also observe in this figure that the ratio for a small α is significantly large, as expected.

In Fig. 2.13, on the other hand, the ratio grows when the number of workers is large. In addition, the ratio for a small β is larger than that for a large β . Note that the event where the worker-processing time is extremely large is likely to occur for a Pareto distribution. Therefore, these results suggest that backup-task scheduling is significantly effective for improving the throughput performance when the event of an extremely-large worker-processing time is likely to occur.

Figure 2.14 (resp. Figure 2.15) illustrates the ratio of the mean response time for Model B to that for Model N in the case of a Weibull (resp. Pareto) distribution. The horizontal axis represents the number of workers in log scale, and the ratio in Fig. 2.14 (resp. Fig. 2.15) is calculated in five cases of α (resp. β).

In Fig. 2.14, the ratio remains almost constant with the increase in the number of workers. In addition, the mean response time for Model N is smaller than that for Model B when $\alpha = 2.000$ and 4.000 . These results suggest that when the system is managed by backup-task scheduling and the worker-processing time follows a Weibull distribution, increasing the number of workers is not significantly effective in improving the response time. Note that when $\alpha = 0.2500, 0.5000$ and 1.000 , the ratio is smaller than one, and this implies that the mean response time for Model B is smaller than that for Model N. Therefore, even in the case of a Weibull distribution, backup-task scheduling can improve the performance when its coefficient of variation is large. In Fig. 2.15, on the other hand, the ratio decreases with the increase in the number of workers. This implies that backup-task scheduling works significantly well in the case of a Pareto distribution. Note that in both figures, the ratio for a large coefficient of variation is significantly small for any number of workers.

Figure 2.16 (resp. Figure 2.17) represents the ratio of the mean total processing time for

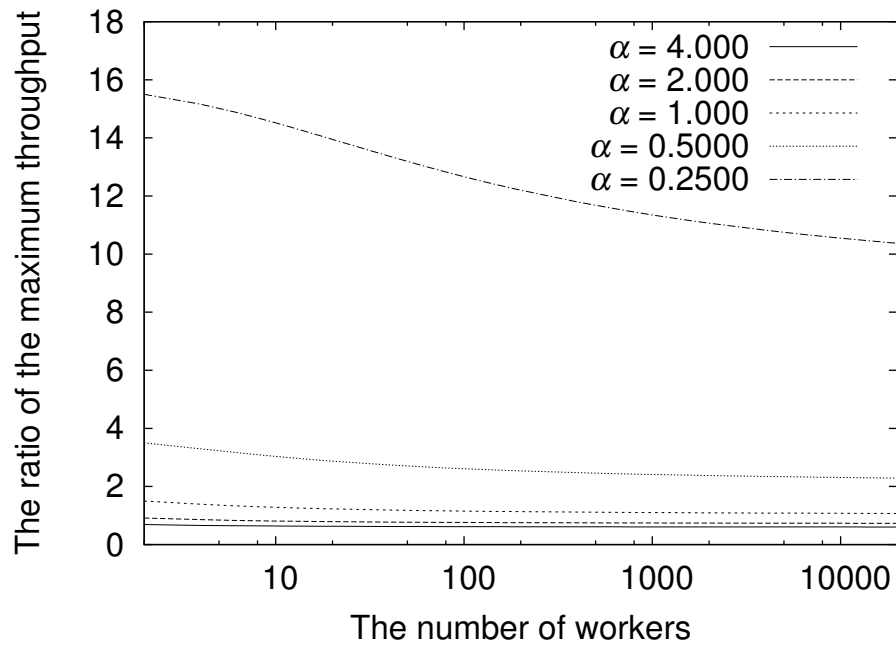


Figure 2.12: Ratio of the maximum throughput for Model B to that for Model N in the case of a Weibull distribution.

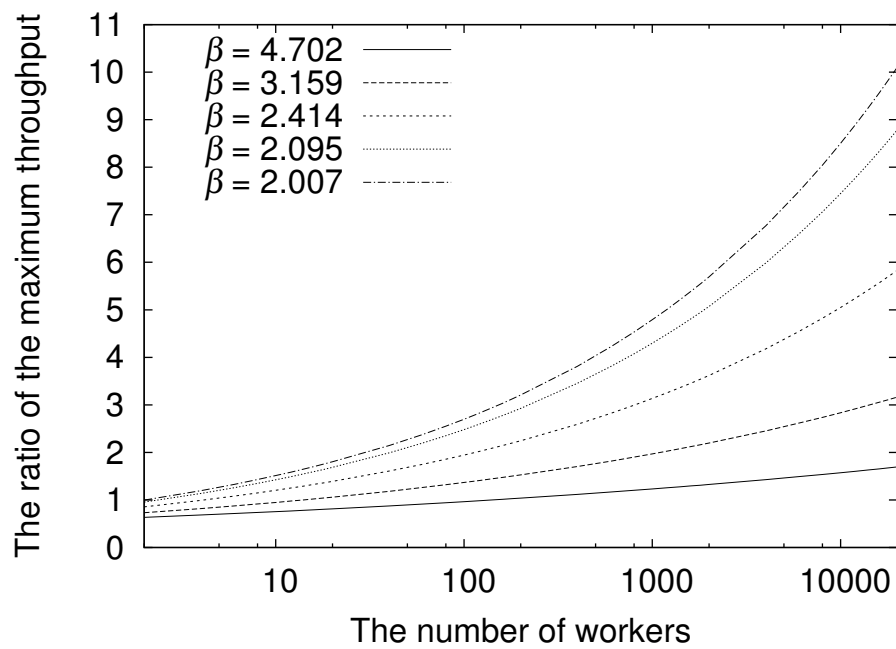


Figure 2.13: Ratio of the maximum throughput for Model B to that for Model N in the case of a Pareto distribution.

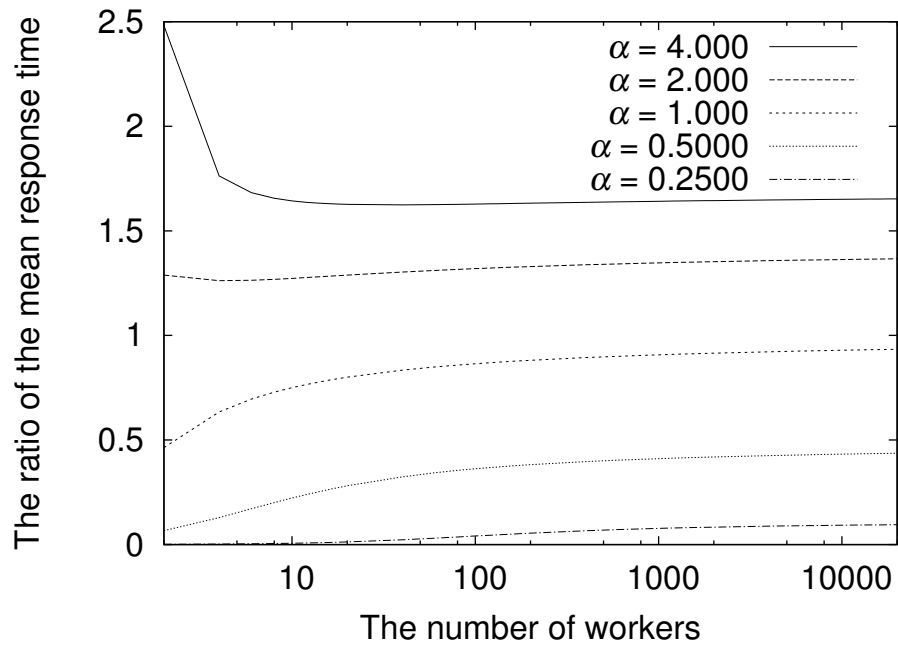


Figure 2.14: Ratio of the mean response time for Model B to that for Model N in the case of a Weibull distribution.

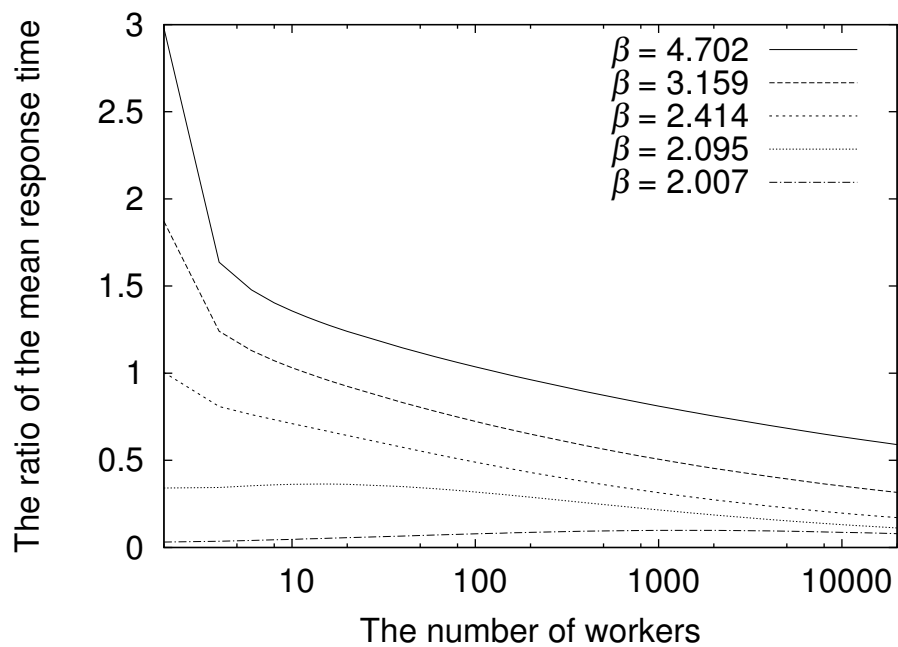


Figure 2.15: Ratio of the mean response time for Model B to that for Model N in the case of a Pareto distribution.

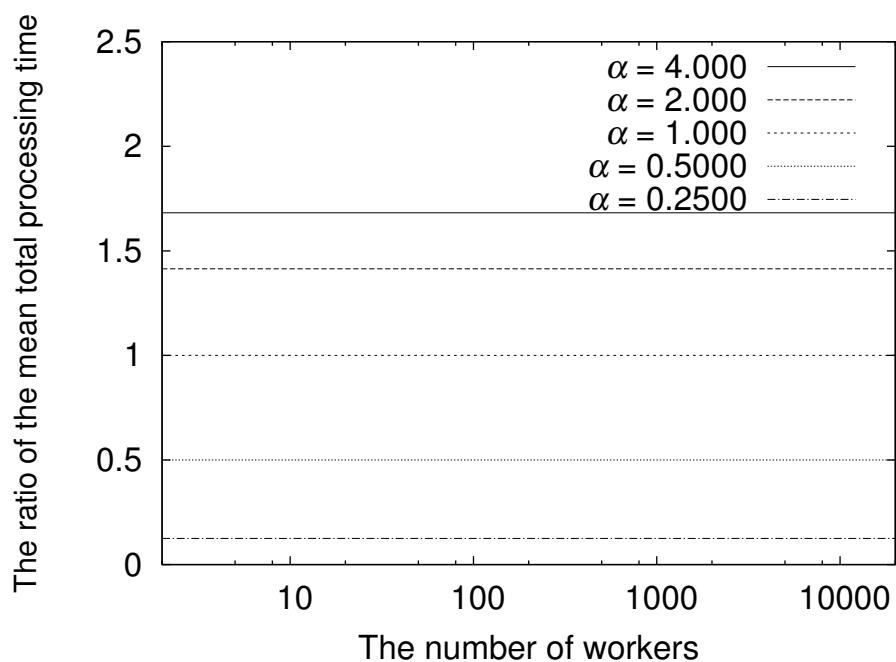


Figure 2.16: Ratio of the mean total processing time for Model B to that for Model N in the case of a Weibull distribution.

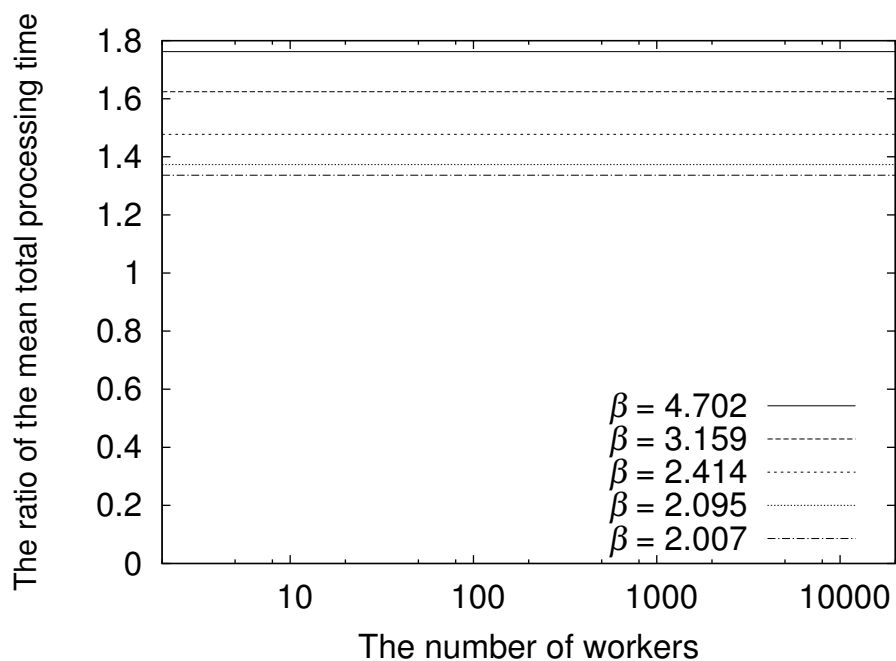


Figure 2.17: Ratio of the mean total processing time for Model B to that for Model N in the case of a Pareto distribution.

Model B to that for Model N in the case of a Weibull (resp. Pareto) distribution. The horizontal axis represents the number of workers in log scale, and the ratio in Fig. 2.16 (resp. Fig. 2.17) is calculated in five cases of α (resp. β). In both figures, the ratio is constant with the increase in the number of workers, and the ratio for a large α (resp. β) is greater than that for a small α (resp. β). That is, backup-task scheduling increases the resource consumption when the variance of the worker-processing time distribution becomes small. Remarkably, in Fig. 2.16, the ratio is less than one for $\alpha < 1$ and the mean total processing time for Model B is less than that for Model N. On the other hand, in Fig. 2.17, the ratio is always greater than one, and backup-task scheduling increases the mean total processing time compared with that of normal scheduling. This implies that backup-task scheduling can reduce the resource consumption when the worker-processing time follows a Weibull distribution with $\alpha < 1$.

In order to conclude this subsection, note first that backup-task scheduling is not effective for a small M because the mean of the worker-processing time distribution for Model B is b/M , which is greater than that for Model N, $b/2M$. Note also that the issue of stragglers rarely occurs for small M . When M is extremely large, the mean of the worker-processing time distribution for Model B is still greater than that for Model N. However, the difference between them is small, and the issue of stragglers is likely to occur when the variance of the worker-processing time distribution is large.

From the numerical results, we can claim that backup-task scheduling is significantly efficient for improving the performance when the variance of the worker-processing time distribution is large. Moreover, the effect of backup-task scheduling depends on the worker-processing time distribution, even though the coefficient of variation is the same. Especially, in the case of many workers, the effect of backup-task scheduling on the maximum throughput and mean response time for a Weibull distribution is very different from that for a Pareto distribution. Therefore, we should pay attention to the shape as well as the first and second order statistics of the worker-processing time distribution when we consider the efficiency of backup-task scheduling.

2.5 Summary

In this chapter, we considered the impact of stragglers and the efficiency of backup-task scheduling in large-scale parallel-distributed processing. We modeled the task-scheduling server as a single-server queue with many workers, deriving the maximum throughput, mean response time and mean total processing time. From the numerical results, we can claim that the issue of stragglers degrades the performance when the coefficient of variation of the worker-processing time is large. Moreover, we can also claim that backup-task scheduling

is significantly efficient for improving the performance in the case of a large variance of the worker-processing time. Note that the effect of backup-task scheduling depends on the shape of the worker-processing time distribution even when the means and variances of the distributions are the same.

Chapter 3

Performance Analysis of Task Replication

3.1 Introduction

In the previous chapter, we consider the impact of stragglers and the efficiency of backup-task scheduling, which is a kind of task-replication scheme. In this chapter, we focus on the effect of task-replication scheduling in terms of the number of replications, and consider how the shape of the worker-processing time distribution affects performance measures.

To this end, we evaluate the effect of task-replication scheduling on two performance measures: the task-processing time and the total amount of execution times of workers for the processing of a task. The latter is referred to as the *total processing time* hereafter. Note that the former indicates how the performance is improved by task replication, whereas the latter characterizes the cost resulting from task replication. We consider the task-replication scheduling policy in which a task entering the service facility is split into subtasks of equal size, and the task service ends when all of the subtasks are completed. Note that the assumption of equally sized subtasks becomes reasonable when a huge amount of input data is split into data pieces of approximately equal size [14]. Moreover, each subtask is processed not only by its own worker but also by alternative distinct workers, and the subtask service ends when one of the relevant workers' processes is completed. As in the previous chapter, we assume that the times to complete subtask by its own worker and alternative distinct workers are independent and identically distributed (i.i.d.), and refer to these times as the *worker-processing times*. Note here that it is reported in [5, 11, 14] that most large distributed systems are heterogeneous and dynamic due to many reasons. For example, a machine with an ill-conditioned disk may suffer from a long disk-read time. The machine-/cluster-level task scheduler may schedule the other tasks before subtasks. Software failure also causes a long worker-processing time. It is also reported that hardware faults and the complexity of software process make the system behavior unpredictable even when the system is operated in a centralized manner. Based on this

unpredictability, we assumed that the worker-processing time is i.i.d. even when the replicated subtasks are the same as the original one. Note also that the subtask-processing time is given by the minimum of some worker-processing times.

For this system, we propose an approach based on extreme value theory for approximately deriving the mean value and standard deviation of the task-processing time. Moreover, we exactly derive the mean total processing time. It is reported in [29, 33] that the time between worker failures has a heavy-tailed property. Therefore, in order to investigate how the tail of the distribution affects performance measures, we consider cases in which the worker-processing time follows a hyper-exponential, Weibull or Pareto distribution¹. These distributions are also used for modeling the worker-processing time in the literature (see, for example, [1, 23, 46]). In numerical examples, we investigate the accuracy of the approximations derived with extreme value theory in comparison with exact analyses. We then determine the optimal number of alternative workers which achieves the shortest task-processing time, and consider the effect of task replication on the performance measures. Finally, we discuss the effect of the starting time of task replication on performance measures through Monte Carlo simulation because we assumed in the analytical model that alternative subtasks are simultaneously executed from the beginning of the processing of the task in order to simplify the analysis.

The remainder of this chapter is organized as follows. We describe previous studies on task replication and point out the differences between these studies and the present study in Section 3.2. The analytical model for large-scale parallel-distributed processing with task replication is described in Section 3.3. For this model, we approximately derive the mean value and standard deviation of the task-processing time using extreme value theory and exactly derive the mean total processing time in Section 3.4. Section 3.5 presents numerical examples of the derived performance measures. Finally, we summarize this chapter in Section 3.6.

3.2 Related Work

A number of studies have investigated the performance of task replication. From the approach based on real-data measurement, Dean et al. [14] implement a task-replication scheme referred to as backup-task scheduling for MapReduce framework and report that backup mech-

¹In this dissertation, we use these distributions according to the measurement of real systems [2, 9, 10]. However, our proposed approach based on extreme value theory can apply to other distributions belonging to the maximum domain of attraction of the extreme value distribution (see Definition A.0.1). Note that the characteristics of the tails of the distributions are similar from the viewpoint of extreme value theory when the distributions are in the maximum domain of attraction of the same extreme value distribution. Therefore, our observation in this chapter can be generalized to some extent for the distributions belonging to the maximum domain of attraction of the same extreme value distribution.

anisms can significantly reduce the task-processing time, increasing computational resource consumption by no more than a few percent. Zaharia et al. [51] focus on a speculative task assignment mechanism, which is a kind of task-replication scheme, implemented on Hadoop [45]. They propose a new task-selection algorithm to improve the accuracy of speculation and confirm through measurement-based evaluation that their algorithm works significantly better than Hadoop's algorithm in heterogeneous environments.

On the other hand, from the viewpoint of simulation experiments, Anglano et al. [3] investigate a scenario in which several users submit multiple sets of tasks to a scheduler simultaneously and propose several set selection strategies while using task replication to process individual sets of tasks. They compare these strategies through discrete-event simulation and confirm the effectiveness of task replication. Cirne et al. [11] investigate the effectiveness of several job-replication schedulers by simulation in comparison with traditional information-based schedulers. In [20], Dobber et al. investigate the effectiveness of dynamic load balancing (DLB) and job replication (JR) by trace-driven simulation experiments and propose a hybrid scheduling scheme of DLB and JR. Nóbrega et al. [38] propose replication schedulers that use any available information about applications and resources and evaluate these schedulers through simulation. They demonstrate that the use of partial information (e.g., the size of the tasks and the speed of the workers) on replication schedulers can greatly decrease resource wastage without affecting the task-processing time.

As mentioned above, many researches investigate the performance of task replication through simulation or measurement. However, these approaches require excessive time or computing resources for evaluating extremely large-scale systems. Moreover, it is difficult to see how the system characteristics, such as the number of workers and the heterogeneity of the worker-processing time, affect performance measures.

To overcome these challenges, the author has investigated the performance of task replication from a theoretical point of view. In Chapter 2, we model the task-scheduling server of parallel-distributed processing as a single-server queue and explicitly derive task-processing time distributions when the worker-processing time obeys a Weibull or Pareto distribution. We then compare the mean response time under task-replication scheduling with the mean response time obtained under normal scheduling, and demonstrate that the effect of task-replication depends significantly on the distribution of the worker-processing time.

Hashimoto et al. [25] consider the effect of backup-task scheduling on the performance of systems with parallel-distributed processing, in which one replicated subtask is activated if an original subtask is not completed by a pre-specified time referred to as the deadline time. They derive approximate formulas for the task-processing time and total processing time by extreme value theory, investigating how the deadline time affects the performance measures for three cases of the distribution of the worker-processing time: hyper-exponential, Weibull and Pareto

distributions. In [25], the number of replications is one, and a primary concern is the effect of the deadline time on the system performance. In the present study, on the other hand, we focus on how the number of replications affects performance measures.

3.3 Analytical Model

We make the following assumptions to construct a stochastic model for large-scale parallel-distributed processing.

- (a) When a task is accepted by the server, the task is divided into N subtasks, each of which is replicated $R - 1$ times.
- (b) The system has a server consisting of M ($:= NR$) workers.
- (c) The M subtasks (N original subtasks and their $N(R - 1)$ copies) are assigned to the M workers on a one-to-one basis (see Figure 3.1).
- (d) The processing of a group consisting of one original subtask and its $R - 1$ copies is terminated when one of the R subtasks is processed completely. Note that there exists some delay for terminating all unfinished subtasks. This delay can be included as a part of the subtask-processing time, but is not taken into consideration in our model due to analytical tractability. Note also that this delay can be negligible if the subtask-processing time is large compared to the overhead of this termination process. We define the subtask-processing time as the processing time of such a group.
- (e) The task-processing time is equal to the maximum of its N subtask-processing times.

As for the time to make replications, the number of replications R is small in general. For example, the default setting of R in Hadoop is three [45]. In such a case, the time to make replications is negligible.

In what follows, we describe the assumption on the subtask-processing time.

Let $U_0^{(i)}$ ($i = 1, 2, \dots, N$) denote the worker-processing time of the i -th original subtask generated from a task. Moreover, for each $i = 1, 2, \dots, N$, let $U_j^{(i)}$ ($j = 1, 2, \dots, R - 1$) denote the worker-processing time of the j -th copy of the i -th original subtask. We now make the following assumption.

- (f) The $U_j^{(i)}$'s ($i = 1, 2, \dots, N, j = 0, 1, \dots, R - 1$) are i.i.d. random variables which follow a common distribution function H_N with positive mean b/N and positive left endpoint $a/N := \inf\{t \in \mathbb{R}; H_N(t) > 0\}$, where $b > a \geq 0$. Note here that the sizes of a task and

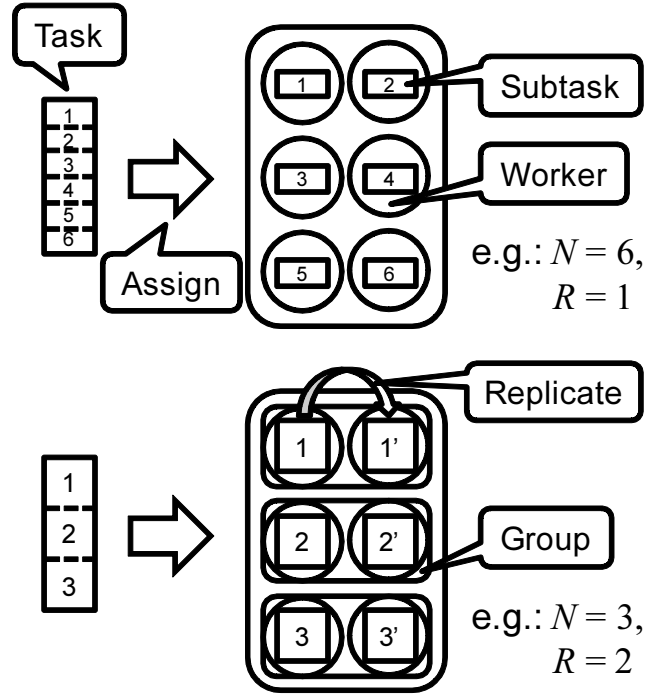


Figure 3.1: Examples of assigning subtasks.

its subtasks are deterministic whereas the processing times of them vary stochastically due to the unpredictability of workers' ability. The distribution H_N is referred to as a *worker-processing time distribution*.

We define S_i ($i = 1, 2, \dots, N$) as the i -th subtask-processing time. It follows from assumption (d) that

$$S_i = \min_{0 \leq j \leq R-1} U_j^{(i)}, \quad i = 1, 2, \dots, N.$$

Clearly, the S_i 's are i.i.d. random variables and

$$F_{N,R}(t) := P(S_i \leq t) = 1 - \{1 - H_N(t)\}^R, \quad t \in \mathbb{R}, \quad (3.1)$$

for all $i = 1, 2, \dots, N$. We also define $T_{N,R}$ as the task-processing time. From assumption (e), we have

$$T_{N,R} = \max_{1 \leq i \leq N} S_i,$$

and thus

$$G_{N,R}(t) := P(T_{N,R} \leq t) = \{F_{N,R}(t)\}^N, \quad t \in \mathbb{R}.$$

3.4 Analysis of Performance Measures

We evaluate the stochastic model described in the previous section in terms of three performance measures.

To this end, we introduce the performance measures. Let $A_{N,R}$ and $D_{N,R}$ denote the mean value and standard deviation of the task-processing time, respectively, i.e.,

$$A_{N,R} = \mathbb{E}[T_{N,R}], \quad D_{N,R} = \sqrt{\text{Var}[T_{N,R}]}.$$

Thus we have

$$A_{N,R} = g_{N,R}^{(1)}, \quad D_{N,R} = \sqrt{g_{N,R}^{(2)} - \left(g_{N,R}^{(1)}\right)^2}, \quad (3.2)$$

where $g_{N,R}^{(k)} = \int_0^\infty t^k dG_{N,R}(t)$ ($k = 1, 2$). Furthermore, let $P_{N,R}$ denote the mean total processing time. Since the total processing time is defined as the total amount of execution times of M workers for the processing of a task, and the S_i 's are i.i.d., we have

$$P_{N,R} = \mathbb{E} \left[\sum_{i=1}^N RS_i \right] = NRf_{N,R}^{(1)}, \quad (3.3)$$

where $f_{N,R}^{(1)} = \int_0^\infty t dF_{N,R}(t)$.

In what follows, we discuss three types of worker-processing time distributions with mean b/N and left endpoint a/N ($b > a \geq 0$). Note here that the aim of introducing left endpoints to the three distributions is to compute performance measures under the same mean and same coefficient of variation for the worker-processing time. This enables us to investigate how the heavy-tailedness of distributions affects the performance measures.

(a) Hyper-exponential distribution²

$$H_N(t) = \begin{cases} 1 - \sigma \exp \{-\sigma(t - a/N)/\nu_N\} \\ \quad - \tilde{\sigma} \exp \{-\tilde{\sigma}(t - a/N)/\nu_N\}, & t \geq a/N, \\ 0, & t < a/N, \end{cases} \quad (3.4)$$

with $\nu_N = (b - a)/(2N)$ and $\tilde{\sigma} = 1 - \sigma$ ($0 < \sigma \leq 1/2$);

(b) Weibull distribution

$$H_N(t) = \begin{cases} 1 - \exp \{-\{(t - a/N)/\eta_N\}^\alpha\}, & t \geq a/N, \\ 0, & t < a/N, \end{cases} \quad (3.5)$$

with $\eta_N = (b - a)/\{\Gamma(1 + 1/\alpha)N\}$ ($\alpha > 0$); and

²Strictly speaking, this is a two-phase balanced hyper-exponential distribution.

(c) Pareto distribution

$$H_N(t) = \begin{cases} 1 - \{\mu_N/(t + \mu_N - a/N)\}^\beta, & t \geq a/N, \\ 0, & t < a/N, \end{cases} \quad (3.6)$$

with $\mu_N = (b - a)(\beta - 1)/N$ ($\beta > 2$).

The coefficients of variation (i.e., the ratio of the standard deviation to the mean) of these distributions are as follows:

(a) Hyper-exponential distribution

$$\frac{b - a}{b} \sqrt{\frac{1}{2\sigma(1 - \sigma)}} - 1; \quad (3.7)$$

(b) Weibull distribution

$$\frac{b - a}{b} \sqrt{\frac{\Gamma(1 + 2/\alpha)}{\{\Gamma(1 + 1/\alpha)\}^2}} - 1; \quad (3.8)$$

(c) Pareto distribution

$$\frac{b - a}{b} \sqrt{\frac{\beta}{\beta - 2}}. \quad (3.9)$$

3.4.1 Exact Expressions for the First and Second Moments of the Task-Processing Time

A straightforward calculation yields the following expressions for $g_{N,R}^{(1)}$ and $g_{N,R}^{(2)}$:

(a) Hyper-exponential distribution

$$\begin{aligned} g_{N,R}^{(1)} &= \frac{1}{N} \sum_{k=1}^N (-1)^{k-1} \binom{N}{k} \sum_{l=0}^{Rk} \binom{Rk}{l} \sigma^l (1 - \sigma)^{Rk-l} \\ &\quad \times \left[\frac{b - a}{2\{\sigma l + (1 - \sigma)(Rk - l)\}} + a \right], \\ g_{N,R}^{(2)} &= \frac{1}{N^2} \sum_{k=1}^N (-1)^{k-1} \binom{N}{k} \sum_{l=0}^{Rk} \binom{Rk}{l} \sigma^l (1 - \sigma)^{Rk-l} \\ &\quad \times \left[\frac{(b - a)^2}{2\{\sigma l + (1 - \sigma)(Rk - l)\}^2} + \frac{a(b - a)}{\sigma l + (1 - \sigma)(Rk - l)} + a^2 \right]; \end{aligned}$$

(b) Weibull distribution

$$g_{N,R}^{(1)} = \frac{1}{N} \sum_{k=1}^N (-1)^{k-1} \binom{N}{k} \left(\frac{b-a}{R^{1/\alpha} k^{1/\alpha}} + a \right),$$

$$g_{N,R}^{(2)} = \frac{1}{N^2} \sum_{k=1}^N (-1)^{k-1} \binom{N}{k} \left[\frac{\Gamma(1+2/\alpha)}{\{\Gamma(1+1/\alpha)\}^2} \frac{(b-a)^2}{R^{2/\alpha} k^{2/\alpha}} + \frac{2a(b-a)}{R^{1/\alpha} k^{1/\alpha}} + a^2 \right];$$

(c) Pareto distribution

$$g_{N,R}^{(1)} = \frac{1}{N} \sum_{k=1}^N (-1)^{k-1} \binom{N}{k} \left\{ \frac{(b-a)(\beta-1)}{\beta R k - 1} + a \right\},$$

$$g_{N,R}^{(2)} = \frac{1}{N^2} \sum_{k=1}^N (-1)^{k-1} \binom{N}{k} \left\{ \frac{2(b-a)^2(\beta-1)^2}{(\beta R k - 2)(\beta R k - 1)} + \frac{2a(b-a)(\beta-1)}{\beta R k - 1} + a^2 \right\}.$$

Combining (3.2) with the above equations, we can obtain *exact* expressions for $A_{N,R}$ and $D_{N,R}$. However, these expressions are not suitable for computing with high accuracy because a number of subtractions in the above equations can cause loss of significant digits. Moreover, it is difficult to make further insights for these expressions against the parameters such as N and R . Therefore, using extreme value theory, we derive asymptotic formulas for $g_{N,R}^{(k)}$ ($k = 1, 2$), which can serve as approximations when N is large. The accuracy of the approximations is numerically investigated in Subsection 3.5.1.

3.4.2 Asymptotic Formulas for the First and Second Moments of the Task-Processing Time

Using extreme value theory, we derive asymptotic formulas for $g_{N,R}^{(1)}$ and $g_{N,R}^{(2)}$ for three cases. Note that the preliminary results of extreme value theory used in this subsection are summarized in appendix A.

Case of a Hyper-Exponential Distribution

Let $F_R(t) = F_{1,R}(t)$. From (3.1), we then have

$$F_R(t) = 1 - \{1 - H_1(t)\}^R, \quad t \in \mathbb{R}. \quad (3.10)$$

Substituting (3.4) into (3.10), we have

$$F_R(t) = \begin{cases} 1 - [\sigma \exp\{-\sigma(t-a)/\nu_1\} - \tilde{\sigma} \exp\{-\tilde{\sigma}(t-a)/\nu_1\}]^R, & t \geq a, \\ 0, & t < a. \end{cases}$$

Note that F_R is independent of N and

$$F_R(t) = F_{N,R}\left(\frac{t}{N}\right) = \mathbb{P}(NS_i \leq t), \quad t \in \mathbb{R}, \quad (3.11)$$

which implies that $\{NS_i; i = 1, 2, \dots, N\}$ is a sequence of i.i.d. random variables with the distribution function F_R . We can confirm that F_R can be expressed as (A.2), where

$$x_0 = a, \quad c(x) = 1, \quad g(x) = 1,$$

$$\begin{aligned} a(x) &= \left[\sigma \exp \left\{ -\frac{\sigma(x-a)}{\nu_1} \right\} + \tilde{\sigma} \exp \left\{ -\frac{\tilde{\sigma}(x-a)}{\nu_1} \right\} \right] \\ &\quad \times R^{-1} \left[\frac{\sigma^2}{\nu_1} \exp \left\{ -\frac{\sigma(x-a)}{\nu_1} \right\} + \frac{\tilde{\sigma}^2}{\nu_1} \exp \left\{ -\frac{\tilde{\sigma}(x-a)}{\nu_1} \right\} \right]^{-1}. \end{aligned}$$

Thus it follows from Proposition A.1.1 that $F_R \in \text{MDA}(\Lambda)$ and $c_n = a(d_n)$, where d_n is given as the solution of the equation $F_R(d_n) = 1 - 1/n$, which can be solved using a numerical method (e.g., Newton's method).

Recall here that $NT_{N,R} = \max_{1 \leq i \leq N} NS_i$ and that the i.i.d. random variables NS_i 's ($i = 1, 2, \dots, N$) follow the distribution function F_R (see (3.11)). Therefore, Proposition A.1.2 implies that

$$\lim_{N \rightarrow \infty} \mathbb{E} \left[\left\{ \frac{NT_{N,R} - d_N}{c_N} \right\}^k \right] = (-1)^k \Gamma^{(k)}(1), \quad k = 1, 2, \quad (3.12)$$

where $\Gamma^{(k)}(1)$ ($k = 1, 2$) is given by (see Subsection 5.4 (ii) in [40])

$$\Gamma^{(1)}(1) = \lim_{x \rightarrow 1} \frac{d}{dx} \Gamma(x) = -\gamma \quad (\gamma : \text{Euler constant}), \quad \Gamma^{(2)}(1) = \lim_{x \rightarrow 1} \frac{d^2}{dx^2} \Gamma(x) = \gamma^2 + \frac{\pi^2}{6}.$$

As a result, from (3.12) and $g_{N,R}^{(k)} = \mathbb{E} [T_{N,R}^k]$ ($k = 1, 2$), we have

$$\begin{aligned} g_{N,R}^{(1)} &\stackrel{N}{\sim} -\Gamma^{(1)}(1) \frac{c_N}{N} + \frac{d_N}{N} = \gamma \frac{c_N}{N} + \frac{d_N}{N}, \\ g_{N,R}^{(2)} &\stackrel{N}{\sim} \Gamma^{(2)}(1) \frac{c_N^2}{N^2} - 2\Gamma^{(1)}(1) \frac{c_N d_N}{N^2} + \frac{d_N^2}{N^2} = \left(\gamma^2 + \frac{\pi^2}{6} \right) \frac{c_N^2}{N^2} + 2\gamma \frac{c_N d_N}{N^2} + \frac{d_N^2}{N^2}, \end{aligned}$$

where $f(x) \stackrel{x}{\sim} g(x)$ represents $\lim_{x \rightarrow \infty} f(x)/g(x) = 1$. Substituting these asymptotic formulas into (3.2), we obtain approximate formulas for $A_{N,R}$ and $D_{N,R}$.

Case of a Weibull Distribution

Substituting (3.5) into (3.10), we have

$$F_R(t) = \begin{cases} 1 - \exp \{-R \{(t - a)/\eta_1\}^\alpha\}, & t \geq a, \\ 0, & t < a. \end{cases}$$

The distribution function F_R has the representation (A.2) with

$$x_0 = a, \quad c(x) = 1, \quad g(x) = 1, \quad a(x) = \frac{\eta_1^\alpha}{\alpha R(x - a)^{\alpha-1}}.$$

Therefore, according to Proposition A.1.1, $F_R \in \text{MDA}(\Lambda)$ and the normalizing constants c_n and d_n are given by

$$c_n = \frac{\eta_1}{\alpha R} \left(\frac{\log n}{R} \right)^{1/\alpha-1}, \quad d_n = \eta_1 \left(\frac{\log n}{R} \right)^{1/\alpha} + a.$$

As a result, Proposition A.1.2 yields

$$\begin{aligned} g_{N,R}^{(1)} &\stackrel{N}{\sim} -\Gamma^{(1)}(1) \frac{c_N}{N} + \frac{d_N}{N} \\ &= \frac{\gamma(b-a)}{\Gamma(1+1/\alpha)\alpha NR} \left(\frac{\log N}{R} \right)^{1/\alpha-1} + \frac{(b-a)}{\Gamma(1+1/\alpha)N} \left(\frac{\log N}{R} \right)^{1/\alpha} + \frac{a}{N}, \quad (3.13) \\ g_{N,R}^{(2)} &\stackrel{N}{\sim} \Gamma^{(2)}(1) \frac{c_N^2}{N^2} - 2\Gamma^{(1)}(1) \frac{c_N d_N}{N^2} + \frac{d_N^2}{N^2} \\ &= \left(\gamma^2 + \frac{\pi^2}{6} \right) \left\{ \frac{(b-a)}{\Gamma(1+1/\alpha)\alpha N} \left(\frac{\log N}{R} \right)^{1/\alpha-1} \right\}^2 \\ &\quad + \frac{2\gamma(b-a)}{\Gamma(1+1/\alpha)\alpha NR} \left(\frac{\log N}{R} \right)^{1/\alpha-1} \left\{ \frac{(b-a)}{\Gamma(1+1/\alpha)N} \left(\frac{\log N}{R} \right)^{1/\alpha} + \frac{a}{N} \right\} \\ &\quad + \left\{ \frac{(b-a)}{\Gamma(1+1/\alpha)N} \left(\frac{\log N}{R} \right)^{1/\alpha} + \frac{a}{N} \right\}^2. \end{aligned}$$

Case of a Pareto Distribution

From (3.6) and (3.10), we have

$$F_R(t) = \begin{cases} 1 - \{\mu_1/(t + \mu_1 - a)\}^{\beta R}, & t \geq a, \\ 0, & t < a. \end{cases}$$

Note that $1 - F_R$ is regularly varying with index $-\beta R$, and thus (A.4) holds. Therefore, Proposition A.2.1 implies that $F_R \in \text{MDA}(\Phi_{\beta R})$, and the normalizing constant c_n is given by

$$c_n = \mu_1 \{n^{1/(\beta R)} - 1\} + a.$$

From Proposition A.2.2, we have

$$\lim_{N \rightarrow \infty} \mathbb{E} \left[\left\{ \frac{NT_{N,R}}{c_N} \right\}^k \right] = \Gamma \left(1 - \frac{k}{\beta R} \right), \quad k = 1, 2.$$

We also obtain the following asymptotic formulas:

$$\begin{aligned} g_{N,R}^{(1)} &\stackrel{N}{\sim} \Gamma \left(1 - \frac{1}{\beta R} \right) \frac{c_N}{N} \\ &= \Gamma \left(1 - \frac{1}{\beta R} \right) \left[\frac{(b-a)(\beta-1)}{N} \{N^{1/(\beta R)} - 1\} + \frac{a}{N} \right], \\ g_{N,R}^{(2)} &\stackrel{N}{\sim} \Gamma \left(1 - \frac{2}{\beta R} \right) \frac{c_N^2}{N^2} \\ &= \Gamma \left(1 - \frac{2}{\beta R} \right) \left[\frac{(b-a)(\beta-1)}{N} \{N^{1/(\beta R)} - 1\} + \frac{a}{N} \right]^2. \end{aligned} \quad (3.14)$$

3.4.3 Exact Expressions for the Mean Total Processing Time

A straightforward calculation yields the following expressions for $f_{N,R}^{(1)}$:

(a) Hyper-exponential distribution

$$f_{N,R}^{(1)} = \frac{1}{N} \sum_{l=0}^R \binom{R}{l} \sigma^l (1-\sigma)^{R-l} \left[\frac{b-a}{2\{\sigma l + (1-\sigma)(R-l)\}} + a \right];$$

(b) Weibull distribution

$$f_{N,R}^{(1)} = \frac{1}{N} \left(\frac{b-a}{R^{1/\alpha}} + a \right);$$

(c) Pareto distribution

$$f_{N,R}^{(1)} = \frac{1}{N} \left\{ \frac{(b-a)(\beta-1)}{\beta R - 1} + a \right\}.$$

Substituting these formulas into (3.3), we obtain exact expressions for $P_{N,R}$, as follows:

(a) Hyper-exponential distribution

$$P_{N,R} = R \sum_{l=0}^R \binom{R}{l} \sigma^l (1-\sigma)^{R-l} \left[\frac{b-a}{2\{\sigma l + (1-\sigma)(R-l)\}} + a \right];$$

(b) Weibull distribution

$$P_{N,R} = R \left(\frac{b-a}{R^{1/\alpha}} + a \right);$$

(c) Pareto distribution

$$P_{N,R} = R \left\{ \frac{(b-a)(\beta-1)}{\beta R - 1} + a \right\}.$$

Table 3.1: Parameter set.

Parameter	Value
b	3.16×10^{10} [sec] (about 1,000 years)
a	2.68×10^{10} [sec] (85% of b)
M	$3 \times 10^5, 3 \times 10^6, 3 \times 10^7$
R	1, 2, 3, 4

Table 3.2: Values of σ , α and β .

Coefficient of variation	0.40	0.30	0.20
σ	0.0676	0.115	0.241
α	0.447	0.548	0.769
β	2.34	2.69	4.73

3.5 Numerical Examples

In this section, we present some numerical examples. We first verify the proposed approximations of the mean value and standard deviation of the task-processing time by comparing the exact expressions and the approximate formulas derived by applying extreme value theory. We then calculate the optimal number of replications required to minimize the mean task-processing time using the derived approximate formulas. We compare the performance measures between the case of the optimal number of replications and that of no-replication and discuss the efficiency of task replication. Moreover, we consider the effect of task replication on reducing the standard deviation of the task-processing time. Finally, we discuss the effect of the starting time of task replication on performance measures through Monte Carlo simulation.

Tables 3.1 and 3.2 show the parameter values used in the numerical experiments. We set these parameters according to the measurement and setting of real systems [15, 39, 45]. The values of σ , α and β are determined such that the coefficient of variation of the worker-processing time distribution takes the values shown in Tab. 3.2. Note that the coefficient of variation for a hyper-exponential (resp. Weibull and Pareto) distribution decreases with the increase in σ (resp. α and β). Note also that the coefficient of variation for each distribution depends on $b - a$, which is the difference between the mean and left endpoint of the distribution (see (3.7) to (3.9)). When $\sigma = 0.500$ in (3.4) (resp. $\alpha = 1.00$ in (3.5)), a hyper-exponential (resp. Weibull) distribution is reduced to a shifted exponential distribution. Under the param-

eter setting of Tab. 3.1, the coefficient of variation of this shifted exponential distribution is equal to 0.15, which is smaller than one, the value of the coefficient of variation for an exponential distribution. Moreover, in the case of the same coefficient of variation, the tail of the hyper-exponential (resp. Pareto) distribution is the lightest (resp. heaviest) among the three distributions (see Figs. 3.2 and 3.3).

3.5.1 Verification of the Proposed Approximations

In this subsection, we investigate the approximation accuracy of the mean value and standard deviation of the task-processing time, while calculating the relative error between the approximations in Subsection 3.4.2 and exact analysis solutions in Subsection 3.4.1.

Figures 3.4, 3.5 and 3.6 show the relative error of the mean task-processing time for $R = 1, 2$, and 3 with respect to the number of original subtasks in a log-log plot. Here, the worker-processing time distribution is set to a hyper-exponential distribution with $\sigma = 0.115$ (resp. a Weibull distribution with $\alpha = 0.548$ and a Pareto distribution with $\beta = 2.69$) in Fig. 3.4 (resp. Figs. 3.5 and 3.6). Figures 3.4 through 3.6 indicate that the relative error tends to decrease with the increase in N , the number of original subtasks. Note here that extreme value theory does not guarantee that the error between the approximations and exact analysis solutions decreases monotonically when N increases. Therefore, as for the case with Fig. 3.4, the relative error can increase when the number of original subtasks is small. However, in particular, in Fig. 3.4 (resp. Figs. 3.5 and 3.6), the relative error is approximately 1.1% (resp. 1.3% and 4.0%) and is sufficiently small when N is 10^4 . This tendency is also observed for other parameter values of the mean value and standard deviation of the task-processing time. These results suggest that approximation formulas from extreme value theory are not accurate but have the relative error smaller than 0.1 when the number of original subtasks is greater than several tens of thousands.

3.5.2 Optimal Number of Replications

In this subsection, we investigate the optimal number of replications which achieves the shortest task-processing time.

Figure 3.7 (resp. Figs. 3.8 and 3.9) shows the optimal number of R for the mean task-processing time when the worker-processing time distribution follows a hyper-exponential (resp. Weibull and Pareto) distribution. The optimal number is calculated by applying the golden section search to the derived approximate formulas. The horizontal axis represents the coefficient of variation (i.e., σ , α , or β are varied). In these figures, the optimal number of replications increases with the increase in the coefficient of variation. At each value of the

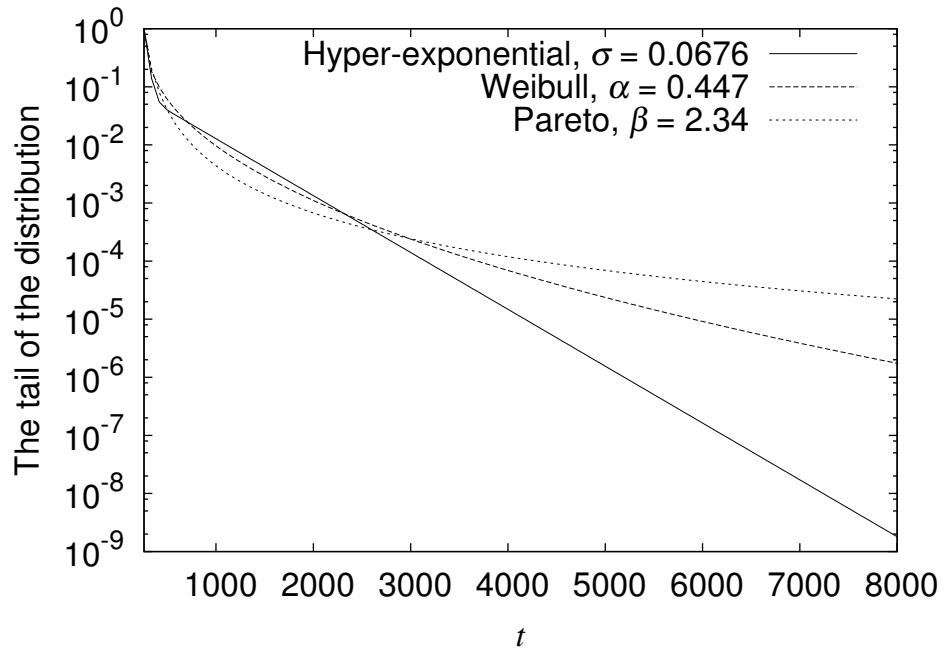


Figure 3.2: Tail of the worker-processing time distribution in the case of a large coefficient of variation.

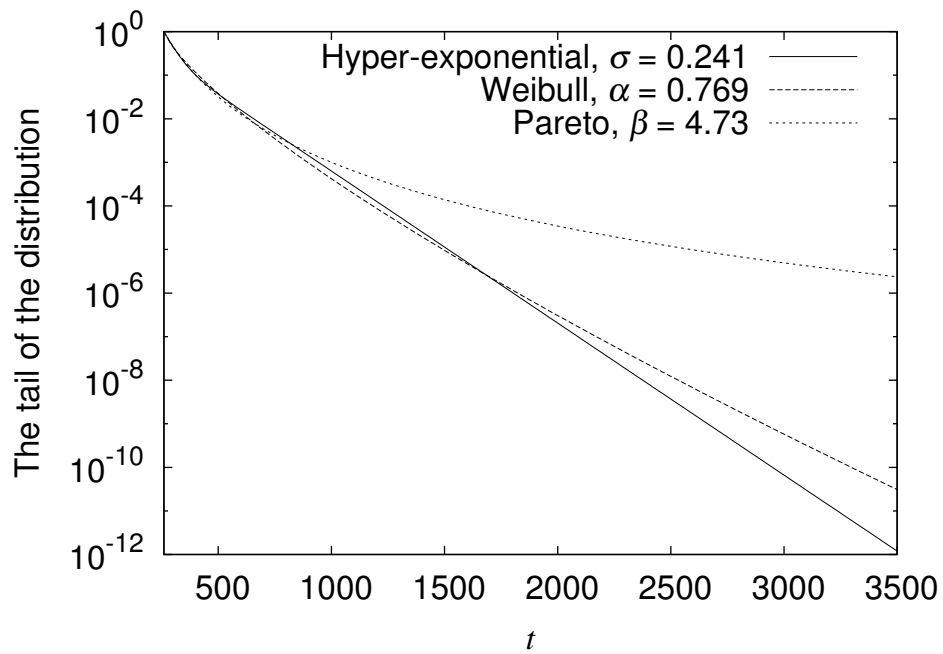


Figure 3.3: Tail of the worker-processing time distribution in the case of a small coefficient of variation.

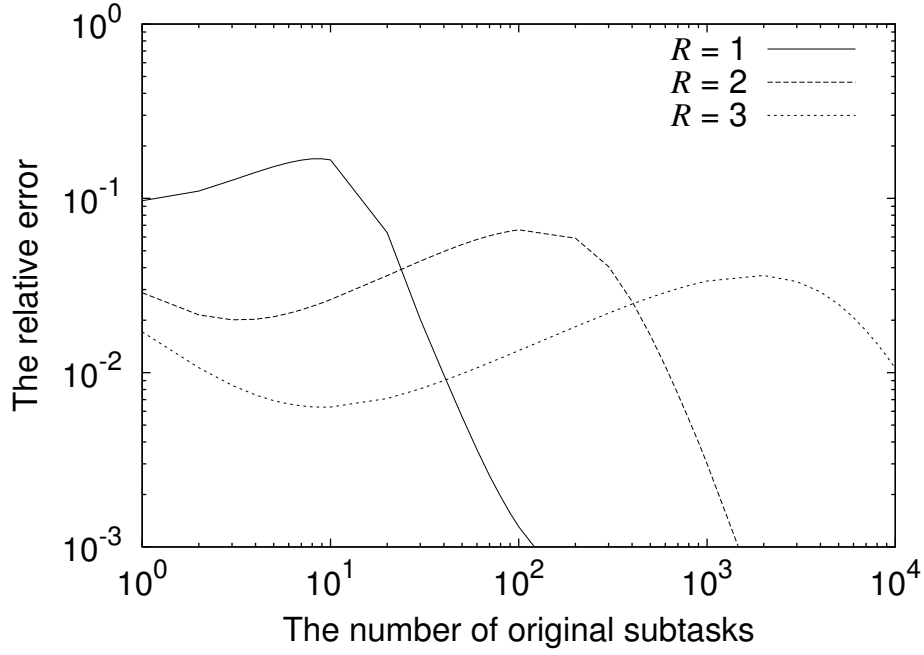


Figure 3.4: Relative error of the mean task-processing time in the case of a hyper-exponential distribution ($\sigma = 0.115$).

coefficient of variation, the optimal number of replications for $M = 3 \times 10^7$ achieves the largest value. However, the difference between optimal values of $M = 3 \times 10^5$ and 3×10^7 is small. There is no significant difference in the optimal number of replications among the three worker-processing time distributions. This implies that the optimal number of replications depends significantly on the variance of the worker-processing time distribution, rather than the tail of distribution.

3.5.3 Efficiency of Task Replication

In this subsection, we discuss the efficiency of task replication by comparing the performance measures between the case of the optimal number of replications and that of no-replication.

Let ${}_h r_{N,R}$ (resp. ${}_w r_{N,R}$ and ${}_p r_{N,R}$) denote the ratio of the mean task-processing time in the case of $R \geq 2$ to that in the case of no-replication (which corresponds to $R = 1$). Note here that the pre-subscript “ h ” (resp. “ w ” and “ p ”) shows that the worker-processing time follows a hyper-exponential (resp. Weibull and Pareto) distribution. By definition,

$${}_h r_{N,R} = \frac{{}_h g_{N,R}^{(1)}}{{}_h g_{N,1}^{(1)}}, \quad {}_w r_{N,R} = \frac{{}_w g_{N,R}^{(1)}}{{}_w g_{N,1}^{(1)}}, \quad {}_p r_{N,R} = \frac{{}_p g_{N,R}^{(1)}}{{}_p g_{N,1}^{(1)}}, \quad (3.15)$$

where ${}_h g_{N,R}^{(1)}$, ${}_w g_{N,R}^{(1)}$ and ${}_p g_{N,R}^{(1)}$ denote the mean task-processing times in the cases of hyper-

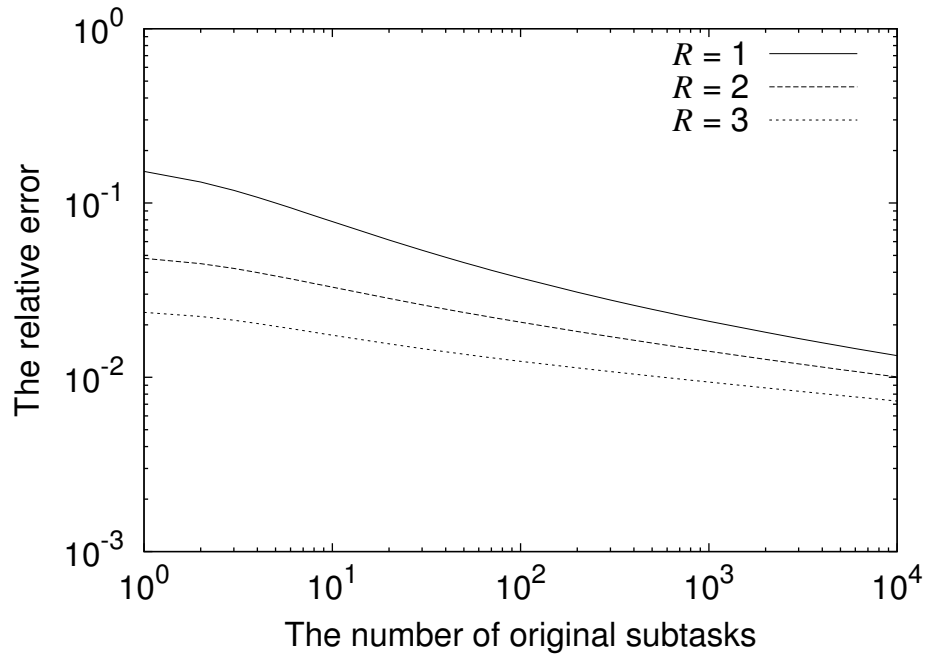


Figure 3.5: Relative error of the mean task-processing time in the case of a Weibull distribution ($\alpha = 0.548$).

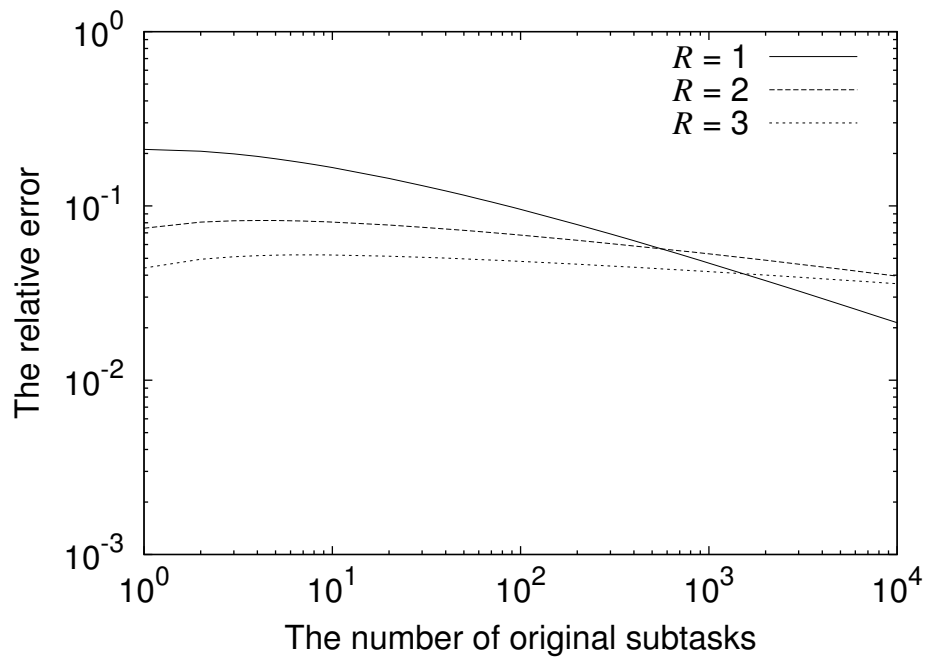


Figure 3.6: Relative error of the mean task-processing time in the case of a Pareto distribution ($\beta = 2.69$).

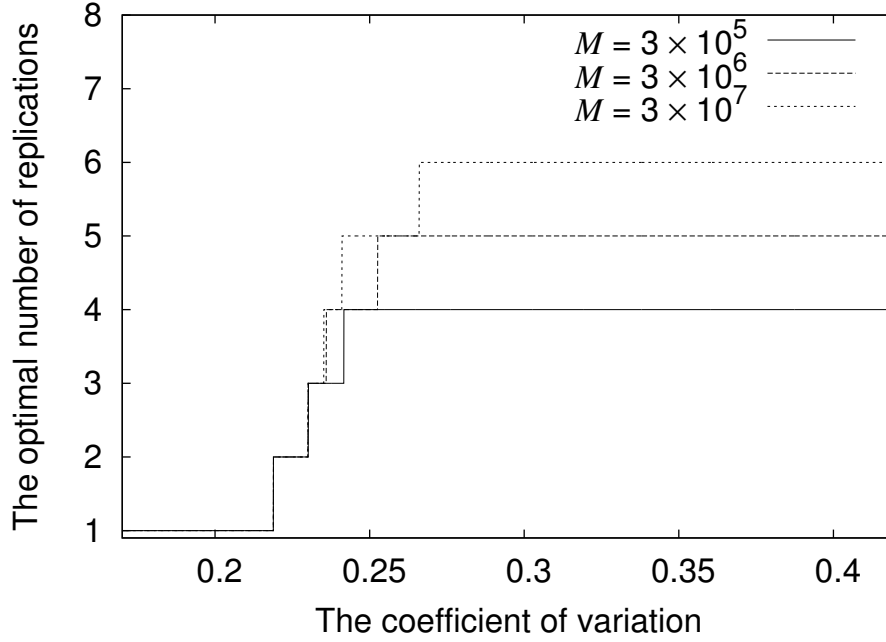


Figure 3.7: Optimal number of replications for the mean task-processing time with respect to the coefficient of variation in the case of a hyper-exponential distribution.

exponential, Weibull and Pareto distributions, respectively. Moreover, we define R^* as the optimal number of replications calculated in Subsection 3.5.2.

Figure 3.10 shows that ${}_h r_{N,R^*}$ gradually decreases with the increase in the coefficient of variation. On the other hand, Figs. 3.11 and 3.12 show that ${}_w r_{N,R^*}$ and ${}_p r_{N,R^*}$ decrease more quickly. Moreover, the value of ${}_h r_{N,R^*}$ for $M = 3 \times 10^5$ is almost the same as the one for 3×10^7 (see Fig. 3.10), whereas the difference between the values of ${}_p r_{N,R^*}$ for $M = 3 \times 10^5$ and $M = 3 \times 10^7$ is comparatively large (see Fig. 3.12). These facts imply that task replication is highly effective in reducing the task-processing time when the variation of the worker-processing time is large, and such effect depends significantly on the tail asymptotics of the worker-processing time distribution.

To verify the above observation from a theoretical point of view, we discuss the difference between ${}_w r_{N,R}$ and ${}_p r_{N,R}$ by using the asymptotic formulas (3.13) and (3.14). Removing the non-dominant terms from these formulas, we have

$${}_w g_{N,R}^{(1)} \sim \frac{(b-a)}{\Gamma(1+1/\alpha)N} \left(\frac{\log N}{R} \right)^{1/\alpha}, \quad (3.16)$$

$${}_p g_{N,R}^{(1)} \sim \Gamma \left(1 - \frac{1}{\beta R} \right) (b-a)(\beta-1)N^{1/(\beta R)-1}. \quad (3.17)$$

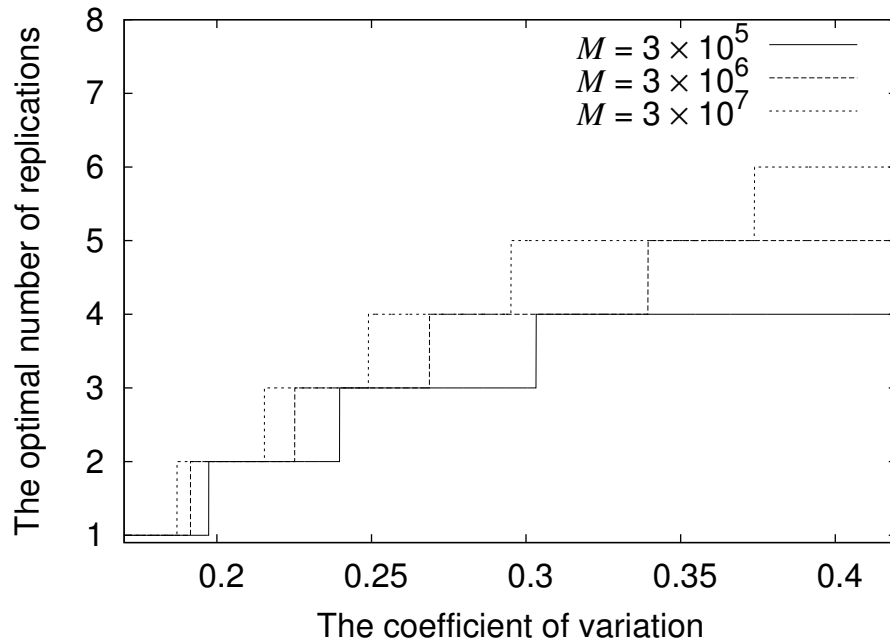


Figure 3.8: Optimal number of replications for the mean task-processing time with respect to the coefficient of variation in the case of a Weibull distribution.

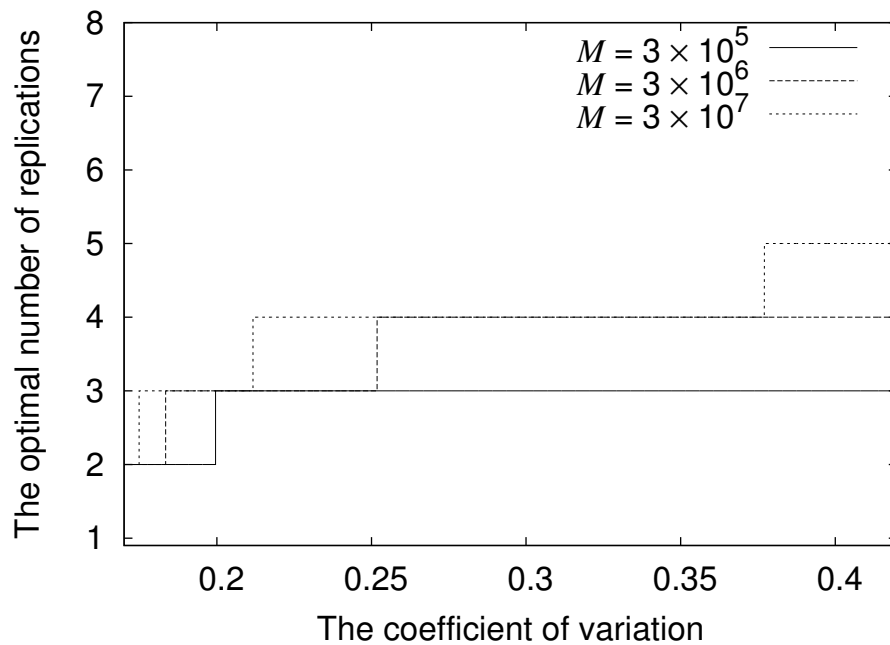


Figure 3.9: Optimal number of replications for the mean task-processing time with respect to the coefficient of variation in the case of a Pareto distribution.

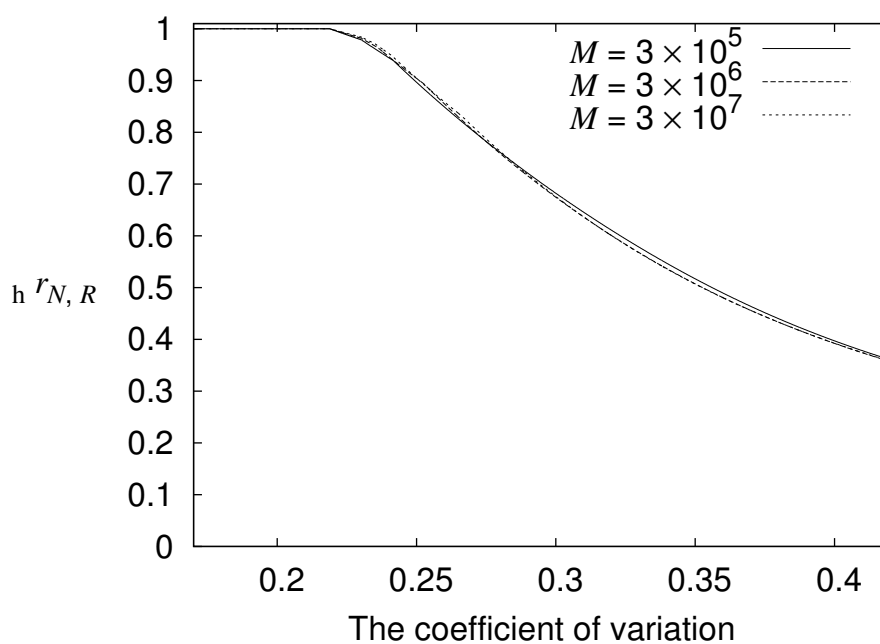


Figure 3.10: Ratio of the mean task-processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a hyper-exponential distribution.

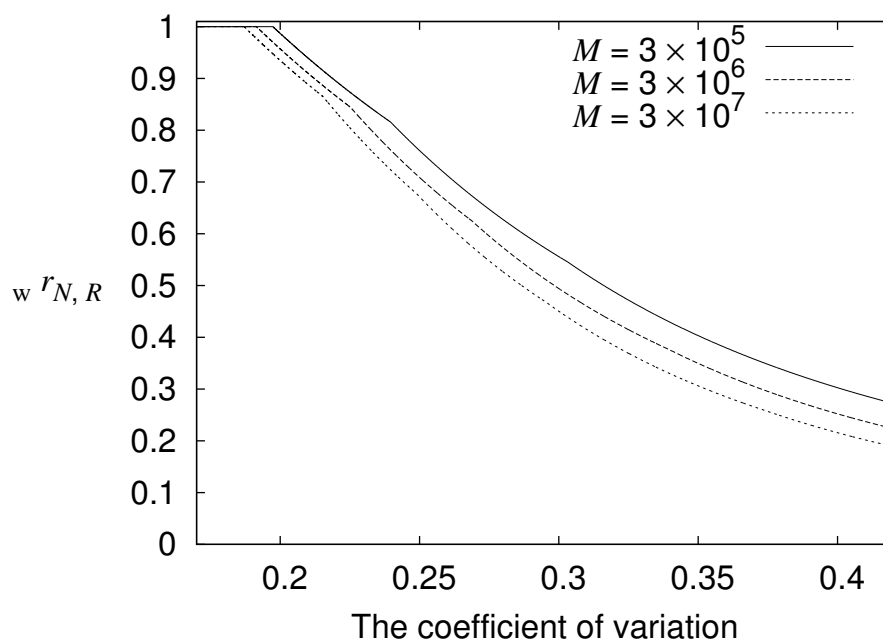


Figure 3.11: Ratio of the mean task-processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Weibull distribution.

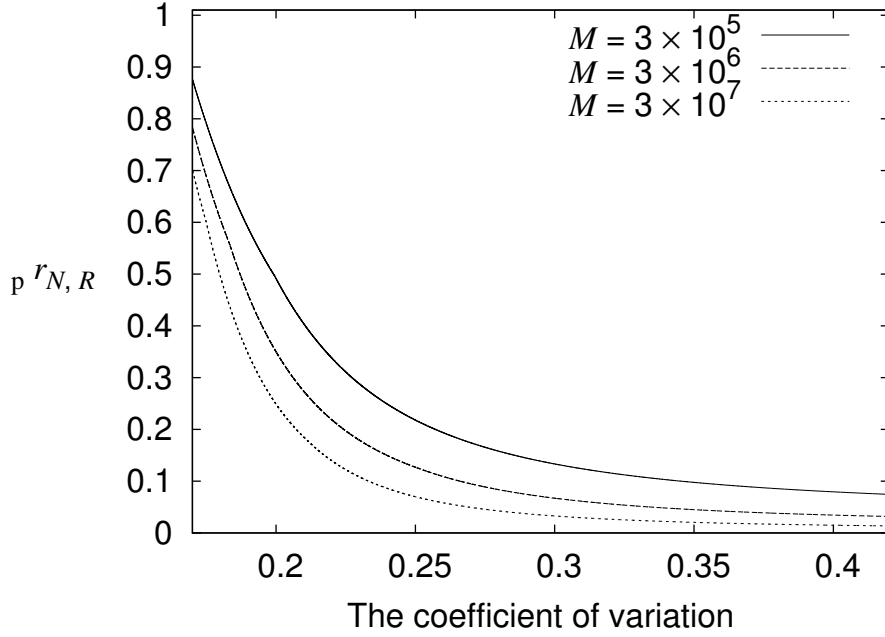


Figure 3.12: Ratio of the mean task-processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Pareto distribution.

Substituting (3.16) and (3.17) into (3.15), we readily obtain, for $R = 2, 3, \dots$,

$$\lim_{N \rightarrow \infty} {}_w r_{N,R} = \left(\frac{1}{R}\right)^{1/\alpha}, \quad \lim_{N \rightarrow \infty} {}_p r_{N,R} = \lim_{N \rightarrow \infty} \frac{\Gamma(1 - 1/(\beta R))}{\Gamma(1 - 1/\beta)} N^{-(1-1/R)/\beta} = 0,$$

which show that

$$\lim_{N \rightarrow \infty} \frac{{}_p r_{N,R}}{{}_w r_{N,R}} = 0. \quad (3.18)$$

Equation (3.18) implies that, as the number of original subtasks is larger, task replication is much more effective in the case of a Pareto distribution, compared to that of a Weibull distribution. This result matches with the observation of Figs. 3.11 and 3.12.

Figure 3.13 (resp. Figs. 3.14 and 3.15) shows the ratio of the mean total processing time in the case of the optimal R to that in the case of $R = 1$ when the worker-processing time distribution follows a hyper-exponential (resp. Weibull and Pareto) distribution. The horizontal axis represents the coefficient of variation. These figures show that the ratios of the mean total processing time increase gradually with the increase in the coefficient of variation. Moreover, the ratios are approximately the same for the three values of the number of workers, which implies that the total processing time depends significantly on the variance of the worker-processing time distribution.

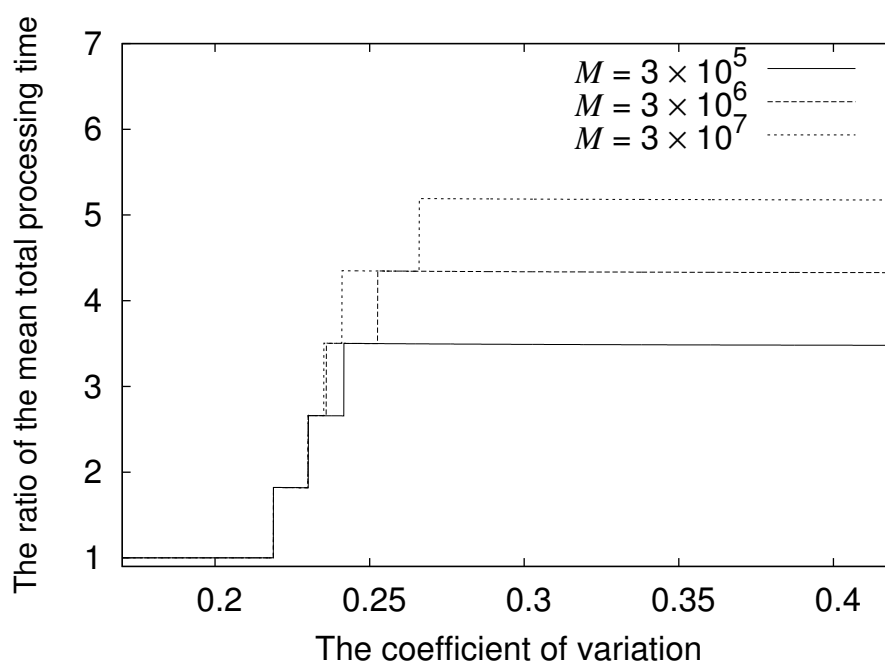


Figure 3.13: Ratio of the mean total processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a hyper-exponential distribution.

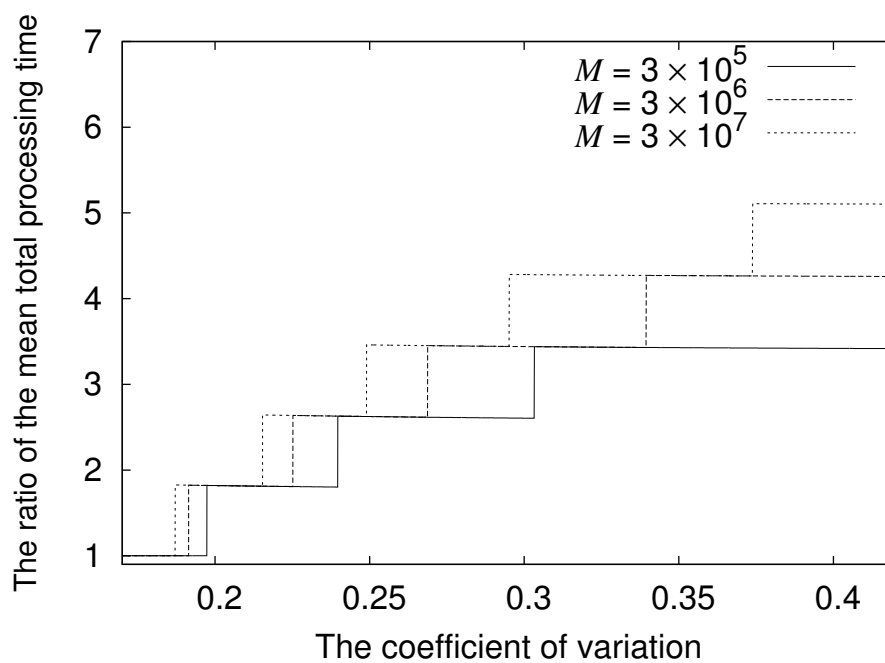


Figure 3.14: Ratio of the mean total processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Weibull distribution.

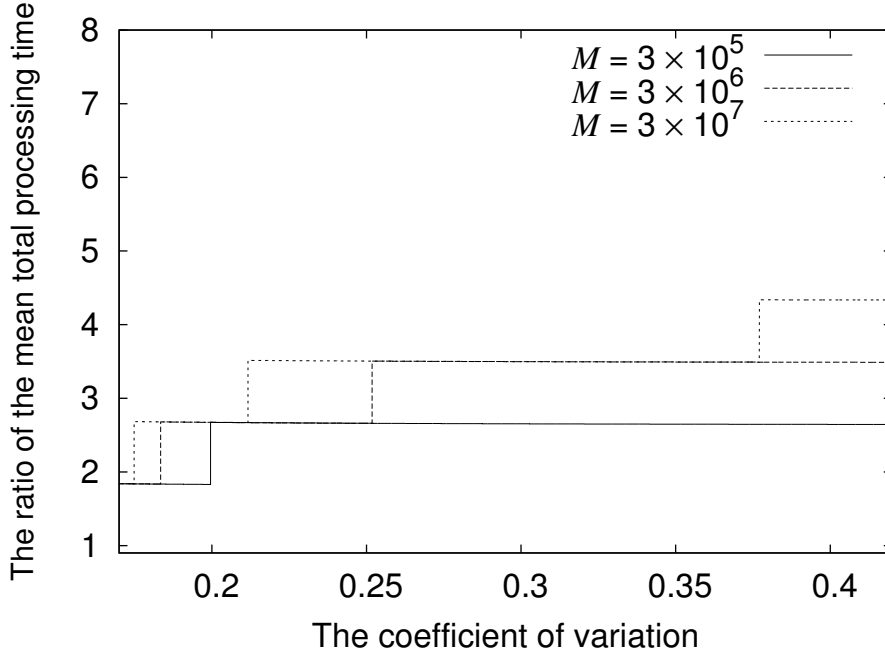


Figure 3.15: Ratio of the mean total processing time for the optimal number of replications to that for no-replication with respect to the coefficient of variation in the case of a Pareto distribution.

Finally, we discuss the effect of task replication on reducing the standard deviation of the task-processing time. Table 3.3 shows the standard deviation in the cases of hyper-exponential, Weibull and Pareto distributions for $M = 3 \times 10^7$.

In Tab. 3.3, we observe that the standard deviation of the task-processing time increases with the increase in the coefficient of variation, and is insensitive to the number of replications in the case of a hyper-exponential distribution. On the other hand, the standard deviation decreases with the number of replications in the cases of Weibull and Pareto distributions. In particular, the case of a Pareto distribution shows a significant decrease in the standard deviation. Note that the standard deviations in the cases of Weibull and Pareto distributions decrease gradually when the number of replications is greater than or equal to three. This implies that task replication is effective for decreasing the variance of the task-processing time. However, a large number of replications is less effective for reducing the variance. Based on numerical experiments, we confirmed that three replications are effective to reduce the variance of the task-processing time, even when the worker-processing time follows a heavy-tailed distribution.

These results indicate that task replication becomes more efficient when the coefficient of variation of the worker-processing time distribution increases. However, this efficiency is

Table 3.3: Standard deviation of the task-processing time.

Distribution	Coefficient of variation	The number of replications			
		1	2	3	4
Hyper-exponential	0.20	4.26×10^2	4.26×10^2	4.25×10^2	4.14×10^2
	0.30	8.92×10^2	8.92×10^2	8.92×10^2	8.92×10^2
	0.40	1.52×10^3	1.52×10^3	1.52×10^3	1.52×10^3
Weibull	0.20	5.38×10^2	4.31×10^2	3.79×10^2	3.46×10^2
	0.30	2.29×10^3	1.25×10^3	8.75×10^2	6.80×10^2
	0.40	6.17×10^3	2.49×10^3	1.46×10^3	1.00×10^3
Pareto	0.20	9.15×10^3	1.19×10^3	6.47×10^2	4.89×10^2
	0.30	1.99×10^5	4.25×10^3	1.51×10^3	9.87×10^2
	0.40	6.75×10^5	6.46×10^3	1.94×10^3	1.19×10^3

very different when the tail of the distribution is changed, and the difference of the mean task-processing time among distributions increases with the increase in the number of workers and the coefficient of variation. Therefore, we should consider the tail of the distribution as well as the first- and second-order statistics of the worker-processing time when we consider the efficiency of task replication for large-scale parallel-distributed processing. In addition, the results also show that three replications are effective to guarantee a low variance of the task-processing time regardless of the tail of the worker-processing time distribution.

3.5.4 Effect of the Starting Time of Task Replication

In this subsection, we discuss the effect of the starting time of task replication through Monte Carlo simulation. In the proposed model, we assumed for analytical simplicity that alternative subtasks are simultaneously executed when the processing of a task starts. In some implementations of task replication [35], on the other hand, backup executions are activated when the elapsed time of the processing of a subtask is greater than a pre-specified threshold. In order to investigate the usefulness of our analytical model, we conduct Monte Carlo simulation experiments. In the simulation setting, backup executions of a subtask start when the elapsed time of the processing of the subtask exceeds $\xi b/M$ ($\xi \geq 0$). Note that the case of $\xi = 0$ corresponds to the setting of the analytical model. We calculated the 95% confidence interval of the mean task-processing time and mean total processing time.

Figure 3.16 shows the mean task-processing time in the case of a Pareto distribution ($\beta = 2.69$, $M = 3 \times 10^5$). The horizontal axis represents ξ . In Fig. 3.16, the mean task-processing time increases more quickly according to ξ in the case of a larger number of replications. For

each R , the analytical result is slightly greater than the simulation result when ξ is small. This discrepancy results from the approximation by extreme value theory. When ξ increases, the simulation result grows linearly, while the analytical result remains constant.

Figure 3.17 shows the mean total processing time in the case of a Pareto distribution ($\beta = 2.69$, $M = 3 \times 10^5$). The horizontal axis indicates ξ . We observe in Fig. 3.17 that the simulation result of the mean total processing time decreases with the increase in ξ , while the analytical result is the same for any ξ .

In Figs. 3.16 and 3.17, the discrepancy between analytical and simulation results is small when ξ is in $[0, 0.1]$. When ξ is greater than 0.1, however, we observe a large discrepancy between them. These trends can be seen for other parameters of hyper-exponential, Weibull and Pareto distributions. This indicates that the analytical model is not applicable when the starting time of task replication is not small. We need further study on formulation and analysis for such a kind of task replication.

3.6 Summary

In this chapter, we evaluated the efficiency of task-replication scheduling in large-scale parallel-distributed processing from a theoretical point of view. To this end, the mean value and standard deviation of the task-processing time were approximately derived with extreme value theory, whereas the mean total processing time was exactly evaluated, in the cases in which the worker-processing time obeys a hyper-exponential, Weibull or Pareto distribution. Through numerical analysis, we verified the proposed approximations by comparing the exact expressions and the approximate formulas derived by applying extreme value theory. We then calculated the optimal number of replications which achieves the shortest task-processing time using the derived approximate formulas. We compared performance measures between the case of the optimal number of replications and that of no-replication. Finally, we considered the effect of task replication on reducing the standard deviation of the task-processing time. We can claim that the efficiency of task-replication scheduling is improved significantly when the coefficient of variation of the worker-processing time increases. However, this efficiency depends significantly on the tail of the worker-processing time distribution even when the means and variances of the distributions are the same. In addition, we can also claim that the optimal number of replications which achieves the shortest task-processing time mainly depends on the coefficient of variation of the worker-processing time. Furthermore, three replications are effective to guarantee a low variance of the task-processing time, regardless of the tail of the worker-processing time distribution.

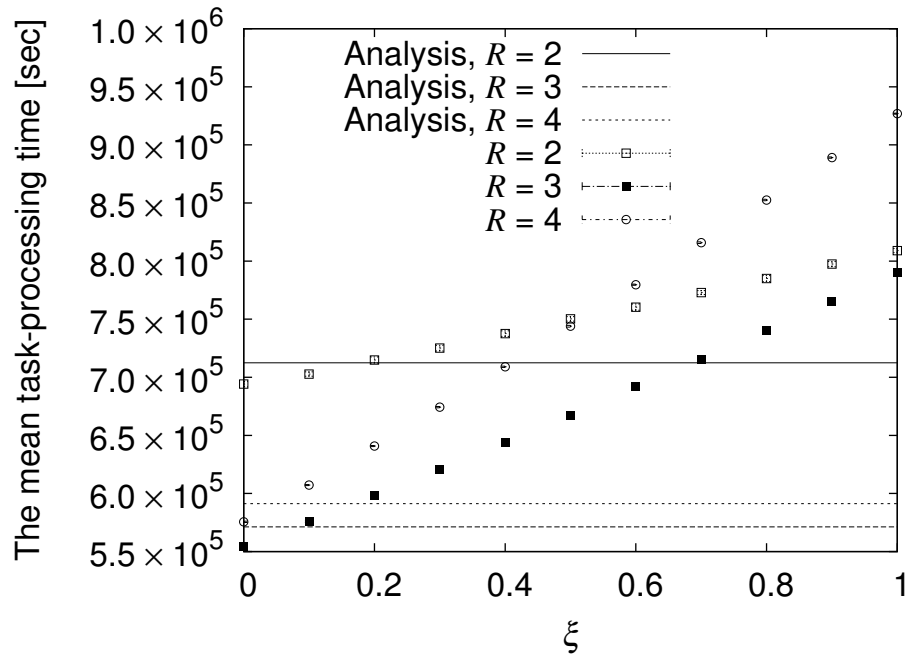


Figure 3.16: Mean task-processing time with simulation in the case of a Pareto distribution ($\beta = 2.69$, $M = 3 \times 10^5$).

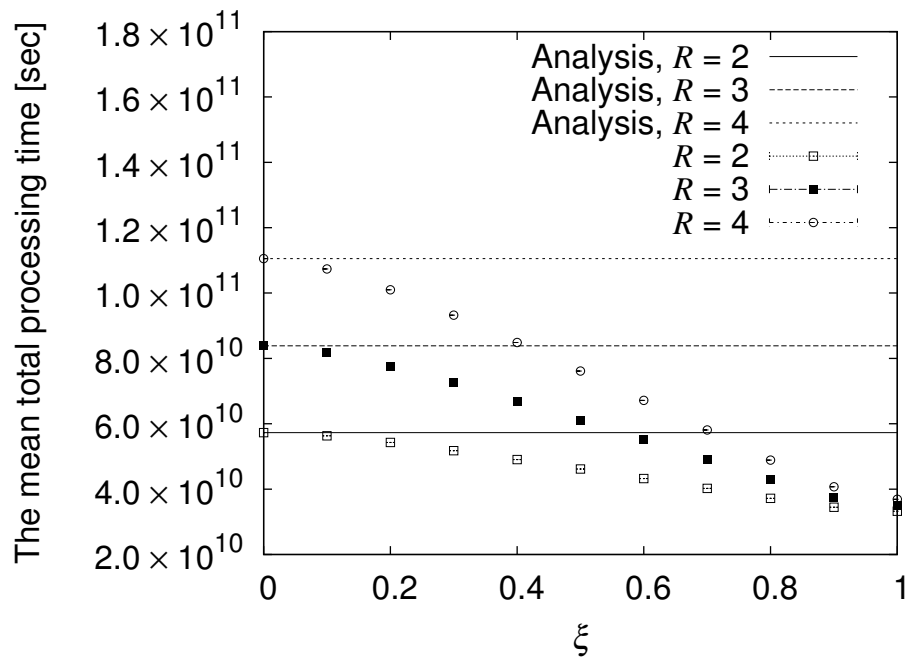


Figure 3.17: Mean total processing time with simulation in the case of a Pareto distribution ($\beta = 2.69$, $M = 3 \times 10^5$).

Chapter 4

Performance Optimization of Checkpointing

4.1 Introduction

Cloud computing, a new usage of computing resources, has rapidly spread in this decade. According to [37], cloud computing is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. These beneficial characteristics enable us to use enormous computing resources conveniently with a low usage fee.

The most successful application framework in cloud computing is a parallel-distributed processing scheme, such as MapReduce [14]. This scheme handles an enormous amount of data with a huge number of computing resources in parallel-distributed processing fashion, and is used for a lot of applications such as data mining, machine learning and social network analysis [7]. As in the previous chapters, we refer to this processing mechanism as *large-scale parallel-distributed processing*.

In large-scale parallel-distributed processing, an enormous task is split into a number of subtasks and those are processed independently in parallel on a cluster of machines referred to as workers. The task completes when all the subtasks have finished, and thus subtasks which are involved in worker failures cause a large processing delay of the task (the issue of stragglers) [14]. Note that, unlike the previous two chapters, we assume here that all the workers process subtasks at the same speed during the processing of a task, and a long time for processing a subtask is caused by re-execution when the dedicated worker fails. Worker failures occur frequently in large-scale parallel-distributed processing because of its huge system scale [16]. For example, it is reported in [15] that there are thousands of hard drive failures for a

new cluster in the first year. In addition, the use of commodity machines for reducing hardware cost decreases the mean time between consecutive failures in data centers [5, 15]. Therefore, dealing with worker failures is an important issue. As in the previous chapters, we refer to the time to complete a task (resp. subtask) as the *task-processing* (resp. *subtask-processing*) time.

Checkpointing [36] is one of the well-known solutions for alleviating the impact of worker failures. In this method, the progress of the processing is periodically saved as a checkpoint during a period of the processing of a subtask. When worker failure occurs, the subtask processed by the failed worker is resumed by another worker from the latest checkpoint. Checkpointing prevents the failed subtask executing from the beginning of the processing and reduces the increase in the subtask-processing time. However, excessive creation of checkpoints increases system overhead, whereas a long checkpoint interval wastes computation resources when worker failure occurs. Therefore, deriving the optimal checkpoint interval is a crucial subject.

In this chapter, we evaluate the effect of checkpointing method on the task-processing time and consider the optimal number of checkpoints which achieves the shortest task-processing time. We construct a stochastic model for large-scale parallel-distributed processing with checkpointing, in which a task accepted by the service facility is split into subtasks of equal size, and the processing of the task ends when all of the subtasks are completed. Note that the assumption of equally sized subtasks becomes reasonable when a huge amount of input data is split into data pieces of approximately equal size [14]. Moreover, each subtask's checkpoints are made periodically during its processing, and a subtask is resumed by other worker from the latest checkpoint when the dedicated worker fails. We assume that the time intervals between consecutive failures of each worker follow an exponential distribution. This distribution is one of the most commonly used distributions due to its tractability and there are various discussions on its validity [6, 18, 30]. We discuss the validity of this assumption in numerical examples.

For this system, we approximately derive explicit expressions for the mean task-processing time and the optimal number of checkpoints. In numerical examples, we investigate the accuracy of the derived approximations in comparison with the results of Monte Carlo simulation. Moreover, we show the usefulness of the derived optimal number of checkpoints by comparing it with the result of previous study. Finally, we validate through Monte Carlo simulation experiments the assumption of an exponential distribution for the time intervals between consecutive worker failures.

The rest of this chapter is organized as follows. Section 4.2 reviews previous studies on checkpointing. In Section 4.3, we illustrate the analytical model for large-scale parallel-distributed processing with checkpointing. For this model, we derive the mean task-processing time and the optimal number of checkpoints in Section 4.4. In Section 4.5, we show numerical

examples of the derived measures. Finally, Section 4.6 summarizes this chapter.

4.2 Related Work

Large-scale parallel-distributed processing is required to minimize the performance degradation due to worker failures. In [12, 36, 52], various implementations of large-scale parallel-distributed processing are proposed, and they apply checkpointing method to reduce the processing delay of a task caused by worker failures. However, none of these studies fully discuss the optimal checkpoint interval.

From the viewpoint of mathematical modeling, Young [49] proposes the model for single processing with checkpointing, and derives the optimal checkpoint interval with a first order approximation. His result is applied to the determination of checkpoint interval not only for single processing but also for parallel-distributed processing [6, 44]. Daly [13] extends Young's result to a higher order approximation, and shows that the optimal checkpoint interval is not affected by the time required to resume a failed task. The results of Young and Daly are simple and insightful. However, it becomes difficult to apply their results to parallel-distributed processing as the system scale grows because their models do not take into account the issue of stragglers.

Fialho et al. [22] and Jin et al. [31] focus on checkpointing method for one of the parallel-distributed processing schemes, Message Passing Interface (MPI), and derive the optimal checkpoint interval. However, in MPI, failed subtask restarts or stops the processing of other subtasks because subtasks are dependent each other by data exchange. The results of the above two studies assume this dependency. On the other hand, in MapReduce, data exchange between subtasks is not required during Map and Reduce steps, and failed subtask does not bother the processing of other subtasks, but becomes a straggler. To the best of the author's knowledge, the effect of checkpointing method for parallel-distributed processing such as MapReduce has not been fully studied yet. In this chapter, therefore, we focus on parallel-distributed processing in which subtasks are processed independently, and derive the optimal number of checkpoints. The derived optimal number of checkpoints is as simple as Young's formula [49], while our formula takes into account the issue of stragglers.

4.3 Analytical Model

In this section, we describe the analytical model consisting of two parts: large-scale parallel-distributed processing and the processing of a subtask with checkpointing.

4.3.1 Model Descriptions of Large-Scale Parallel-Distributed Processing

We describe a stochastic model for large-scale parallel-distributed processing. When a task is accepted by the server, the task is divided into M ($M = 1, 2, \dots$) subtasks, and each subtask is assigned to a dedicated worker. Let S_i ($i = 1, 2, \dots, M$) denote the subtask-processing time of the i -th subtask. The subtask-processing times $\{S_i; i = 1, 2, \dots, M\}$ are independent and identically distributed (i.i.d.) random variables which follow a common distribution function F , i.e.,

$$P(S_i \leq t) = F(t), \quad t \in \mathbb{R}.$$

We refer to the distribution F as the *subtask-processing time distribution*.

We now define T as the task-processing time, which is equal to the maximum of M subtask-processing times, i.e.,

$$T = \max_{1 \leq i \leq M} S_i.$$

We also define G and $g^{(1)}$ as the distribution function and mean value of the task-processing time T . It then follows that

$$\begin{aligned} G(t) &= P(T \leq t) = \{F(t)\}^M, \quad t \in \mathbb{R}, \\ g^{(1)} &= E[T] = \int_0^\infty t dG(t) = \int_0^\infty t d\{F(t)\}^M. \end{aligned} \quad (4.1)$$

In what follows, we refer to $g^{(1)}$ as the *mean task-processing time*.

4.3.2 Model Descriptions of the Processing of a Subtask with Checkpointing

We introduce some symbols and assumptions to describe the processing of a subtask with checkpointing (see Fig. 4.1). That is, the subtask-processing time S_i ($i = 1, 2, \dots, M$) follows these assumptions.

- (a) We define b ($b > 0$) as the subtask-processing time without checkpointing and worker failures, i.e., the actual processing time of a subtask. We also define c ($c > 0$) as the time required to make one checkpoint. Moreover, let K ($K = 0, 1, \dots$) denote the number of checkpoints being made during a period of the processing of a subtask.
- (b) The processing work of a subtask is split into $K + 1$ segments of equal size, and a checkpoint is created at the end of each segment except the last one. Let σ_k ($k = 1, 2, \dots, K + 1$) denote the k -th time interval including the processing time of the k -th

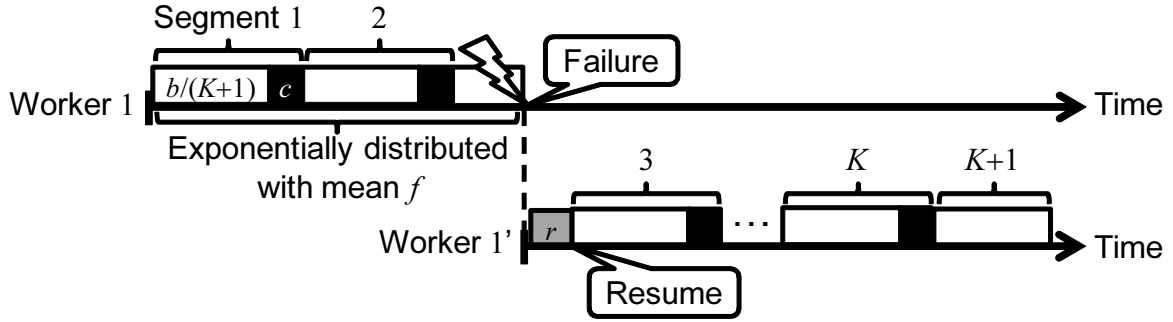


Figure 4.1: Processing of a subtask with checkpointing method.

segment and the time of creating a checkpoint (if any). We then have

$$\sigma_k = \begin{cases} b/(K+1) + c =: \sigma, & k = 1, 2, \dots, K, \\ b/(K+1) =: \sigma', & k = K+1. \end{cases} \quad (4.2)$$

We refer to σ_k as the k -th *segment-processing time*. Let τ denote the subtask-processing time with K checkpoints and no worker failures. We then have

$$\tau = \sum_{k=1}^{K+1} \sigma_k = K\sigma + \sigma' = b + Kc. \quad (4.3)$$

- (c) The time intervals between consecutive failures for each worker are i.i.d. with an exponential distribution having mean f ($f > 0$). When worker failure occurs during a segment-processing time, the subtask involved in failure is resumed by another worker from the checkpoint recorded just before the segment-processing time. We here define r ($r \geq 0$) as the time required to resume the failed subtask.

- (d) The magnitudes of the parameters b, c, K, f and r satisfy

$$Kc, r \ll b \ll f, \quad (4.4)$$

which is a typical relation for parallel-distributed processing with checkpointing on large-scale computing resources [6, 15].

We note that from these assumptions, the subtask-processing time is given as the sum of b , Kc , the total computation time lost by worker failures and the total time required to resume the failed subtask.

4.4 Analysis

In this section, we first consider the worker failure probability and the processing delay (defined in Subsections 4.4.1 and 4.4.2, respectively). We then derive the subtask-processing time distribution F and the mean task-processing time $g^{(1)}$. Finally, we propose the optimal number of checkpoints which minimizes $g^{(1)}$. In the analysis, in order to express the optimal number of checkpoints as a simple formula, we apply some approximations.

4.4.1 Worker Failure Probability

For simplicity, we refer to an arbitrarily chosen subtask as a *tagged subtask*. Let p denote the probability that the tagged subtask is involved in worker failures during its processing, i.e., at least one worker failure happens in the processing of the tagged subtask, which is referred to as the *worker failure probability*. We assume that the processing of the tagged subtask begins at time 0. We then define $T_{\text{fail}}^{(j)}$ ($j = 1, 2, \dots$) as the time at which the j -th worker failure happens to the tagged subtask. It follows from assumption (c) that the time intervals between consecutive worker failures, $T_{\text{fail}}^{(1)}$, $T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)}$, $T_{\text{fail}}^{(3)} - T_{\text{fail}}^{(2)}$, $\dots$, are i.i.d. with an exponential distribution having mean f . We also define N_{fail} as the number of worker failures happening to the tagged subtask during its processing. From these definitions, we have

$$p = \mathbb{P}(N_{\text{fail}} \geq 1) = \mathbb{P}\left(T_{\text{fail}}^{(1)} \leq \tau\right) = 1 - \exp\left\{-\frac{\tau}{f}\right\} = 1 - \exp\left\{-\frac{b + Kc}{f}\right\}, \quad (4.5)$$

where the last equality is due to (4.3). Note here that (4.4) implies $b + Kc \ll f$ and thus $(b + Kc)/f \ll 1$. Therefore, using the linear approximation, the worker failure probability p in (4.5) can be estimated in the following way.

$$p \approx \frac{b + Kc}{f} \ll 1. \quad (4.6)$$

Next, we estimate the probability $\mathbb{P}(N_{\text{fail}} \geq 2)$.

$$\begin{aligned} \mathbb{P}(N_{\text{fail}} \geq 2) &= \sum_{k=1}^{K+1} \mathbb{P}\left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} \geq 2\right) \\ &= \sum_{k=1}^{K+1} \mathbb{P}\left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\right) \mathbb{P}\left(N_{\text{fail}} \geq 2 \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\right), \end{aligned} \quad (4.7)$$

where

$$\tau_k = \begin{cases} k\sigma, & k = 0, 1, \dots, K, \\ K\sigma + \sigma' = \tau, & k = K + 1. \end{cases} \quad (4.8)$$

Recall here (see assumption (c)) that $T_{\text{fail}}^{(1)}$ and $T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)}$ are i.i.d. with an exponential distribution having mean f , and the event $\{N_{\text{fail}} \geq 2\}$ is equivalent to the event $\{T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} < r + \tau - \tau_{k-1}\}$, given that $\{\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\}$. Based on these facts, we obtain

$$\begin{aligned}
 \mathbb{P}\left(N_{\text{fail}} \geq 2 \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\right) &= \mathbb{P}\left(T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} < r + \tau - \tau_{k-1} \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\right) \\
 &= \mathbb{P}\left(T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} < r + \tau - \tau_{k-1}\right) \\
 &\leq \mathbb{P}\left(T_{\text{fail}}^{(2)} - T_{\text{fail}}^{(1)} < r + \tau\right) \\
 &= 1 - \exp\left\{-\frac{r + \tau}{f}\right\} \\
 &= 1 - \exp\left\{-\frac{r + b + Kc}{f}\right\} =: p',
 \end{aligned} \tag{4.9}$$

where we use (4.3) in the last but one equality. Substituting (4.5) and (4.9) into (4.7) yields

$$\mathbb{P}(N_{\text{fail}} \geq 2) \leq p' \sum_{k=1}^{K+1} \mathbb{P}\left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k\right) = p' \mathbb{P}\left(T_{\text{fail}}^{(1)} < \tau\right) = p'p. \tag{4.10}$$

In addition, proceeding as in the derivation of (4.6), we can readily obtain

$$p' \approx \frac{r + b + Kc}{f} \ll 1. \tag{4.11}$$

Applying (4.6) and (4.11) to (4.10) leads to

$$\mathbb{P}(N_{\text{fail}} \geq 2) \ll p \ll 1,$$

and thus

$$\mathbb{P}(N_{\text{fail}} = 0) = 1 - p, \quad \mathbb{P}(N_{\text{fail}} = 1) \approx p. \tag{4.12}$$

In the rest of this section, we assume that the probability $\mathbb{P}(N_{\text{fail}} \geq 2)$ is negligible.

4.4.2 Processing Delay

We consider the processing delay of the tagged subtask caused by worker failure, under the assumption that $N_{\text{fail}} = 1$. The *processing delay* is defined as the sum of the time required to resume a failed subtask and the computation time lost by worker failure, i.e., the time between completion of creating the latest checkpoint and worker failure. Given that $\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k$ ($k = 1, 2, \dots, K + 1$) and $N_{\text{fail}} = 1$, the processing delay is equal to $r + T_{\text{fail}}^{(1)} - \tau_{k-1}$. It then

follows from assumption (c) that

$$\begin{aligned}
& \mathbb{P} \left(r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1 \right) \\
&= \frac{\mathbb{P} \left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < t - r + \tau_{k-1}, N_{\text{fail}} = 1 \right)}{\mathbb{P} \left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1 \right)} \\
&= \frac{\mathbb{P} \left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < t - r + \tau_{k-1} \right)}{\mathbb{P} \left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \right)} \\
&= \frac{(1 - \exp\{-(t - r + \tau_{k-1})/f\}) - (1 - \exp\{-\tau_{k-1}/f\})}{(1 - \exp\{-\tau_k/f\}) - (1 - \exp\{-\tau_{k-1}/f\})} \\
&= \frac{1 - \exp\{-(t - r)/f\}}{1 - \exp\{-\sigma/f\}}, \quad k = 1, 2, \dots, K, \quad r \leq t < r + \sigma, \quad (4.13)
\end{aligned}$$

where the last equality is due to $\tau_k = \tau_{k-1} + \sigma$ (see (4.8)). Similarly, for $r \leq t < r + \sigma'$,

$$\mathbb{P} \left(r + T_{\text{fail}}^{(1)} - \tau_K \leq t \mid \tau_K \leq T_{\text{fail}}^{(1)} < \tau, N_{\text{fail}} = 1 \right) = \frac{1 - \exp\{-(t - r)/f\}}{1 - \exp\{-\sigma'/f\}}. \quad (4.14)$$

Note here that (4.2) and (4.4) imply $c \ll b/(K+1)$ and $\sigma = b/(K+1) + c \ll f$, and using the linear approximation, we have

$$\sigma' = \frac{b}{K+1} \approx \frac{b}{K+1} + c = \sigma, \quad 1 - \exp\left\{-\frac{t}{f}\right\} \approx \frac{t}{f}, \quad 0 \leq t < \sigma. \quad (4.15)$$

Applying (4.15) to (4.13) and (4.14), we have, for $k = 1, 2, \dots, K+1$ and $r \leq t < r + \sigma$,

$$\mathbb{P} \left(r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1 \right) \approx \frac{t - r}{\sigma}. \quad (4.16)$$

4.4.3 Mean Task-Processing Time

We first consider the subtask-processing time distribution F . Let S denote the subtask-processing time of the tagged subtask. From (4.12), we have

$$\begin{aligned}
F(t) &= \mathbb{P}(S \leq t) \\
&= \mathbb{P}(N_{\text{fail}} = 0)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) + \mathbb{P}(N_{\text{fail}} = 1)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 1) \\
&\quad + \mathbb{P}(S \leq t, N_{\text{fail}} \geq 2) \\
&\approx (1 - p)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) + p\mathbb{P}(S \leq t \mid N_{\text{fail}} = 1) \\
&= (1 - p)\mathbb{P}(S \leq t \mid N_{\text{fail}} = 0) \\
&\quad + p \sum_{k=1}^{K+1} \mathbb{P} \left(S \leq t \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1 \right) \\
&\quad \times \mathbb{P} \left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \mid N_{\text{fail}} = 1 \right). \quad (4.17)
\end{aligned}$$

Note here that

$$P(S \leq t \mid N_{\text{fail}} = 0) = \begin{cases} 0, & 0 \leq t < \tau, \\ 1, & t \geq \tau. \end{cases} \quad (4.18)$$

Note also that $S = \tau + r + T_{\text{fail}}^{(1)} - \tau_{k-1}$ for $k = 1, 2, \dots, K + 1$, given that $\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k$ and $N_{\text{fail}} = 1$. Combining this fact and (4.16), we have

$$\begin{aligned} & P\left(S \leq t \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1\right) \\ &= P\left(r + T_{\text{fail}}^{(1)} - \tau_{k-1} \leq t - \tau \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1\right) \\ &\approx \frac{t - \tau - r}{\sigma}, \quad k = 1, 2, \dots, K + 1, \quad \tau + r \leq t < \tau + r + \sigma, \end{aligned}$$

and thus

$$\begin{aligned} & \sum_{k=1}^{K+1} P\left(S \leq t \mid \tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k, N_{\text{fail}} = 1\right) P\left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \mid N_{\text{fail}} = 1\right) \\ &\approx \frac{t - \tau - r}{\sigma} \sum_{k=1}^{K+1} P\left(\tau_{k-1} \leq T_{\text{fail}}^{(1)} < \tau_k \mid N_{\text{fail}} = 1\right) \\ &= \frac{t - \tau - r}{\sigma} P\left(T_{\text{fail}}^{(1)} < \tau \mid N_{\text{fail}} = 1\right) \\ &= \frac{t - \tau - r}{\sigma}, \quad \tau + r \leq t < \tau + r + \sigma. \end{aligned} \quad (4.19)$$

Applying (4.18) and (4.19) to (4.17) yields

$$F(t) \approx \begin{cases} (1 - p), & \tau \leq t < \tau + r, \\ (1 - p) + p(t - \tau - r)/\sigma, & \tau + r \leq t < \tau + r + \sigma. \end{cases} \quad (4.20)$$

Next, we derive an approximation of the mean task-processing time $g^{(1)}$. Substituting (4.20) into (4.1) and using the binomial theorem, we have

$$g^{(1)} \approx \tau + r \{1 - (1 - p)^M\} + Mp\sigma \sum_{i=0}^{M-1} \binom{M-1}{i} \frac{p^i (1 - p)^{M-1-i}}{i + 2}. \quad (4.21)$$

Note here that on the right-hand side of (4.21), the second and third terms mean the expected values of the time required to resume a failed subtask and the computation time lost by worker failure, respectively. We now define $R(z)$ as

$$R(z) = \sum_{i=0}^{M-1} \binom{M-1}{i} p^i (1 - p)^{M-1-i} z^{i+1} = z(1 - p + pz)^{M-1}.$$

It then follows from (4.21) that

$$\begin{aligned} g^{(1)} &\approx \tau + r \{1 - (1 - p)^M\} + Mp\sigma \int_0^1 R(z)dz \\ &= \tau + r \{1 - (1 - p)^M\} + \sigma \left\{1 - \frac{1 - (1 - p)^{M+1}}{(M + 1)p}\right\}. \end{aligned}$$

Substituting the definitions (4.2), (4.3) and (4.5) of σ , τ and p , we have

$$\begin{aligned} g^{(1)} &\approx b + Kc + r \left\{1 - \left(\exp \left\{-\frac{b + Kc}{f}\right\}\right)^M\right\} \\ &+ \left(\frac{b}{K + 1} + c\right) \left\{1 - \frac{1 - (\exp\{-(b + Kc)/f\})^{M+1}}{(M + 1)(1 - \exp\{-(b + Kc)/f\})}\right\} =: \tilde{g}^{(1)}(K). \end{aligned} \quad (4.22)$$

The accuracy of the approximation (4.22) is numerically investigated in Subsection 4.5.1.

4.4.4 Optimal Number of Checkpoints

In this subsection, we show a simple and explicit formula for the *optimal number* of checkpoints which is expected to minimize the mean task-processing time $g^{(1)}$. To this end, we simplify the approximate expression (4.22) of $g^{(1)}$. From (4.4), we have $(b + Kc)/f \ll 1$ and $Kc \ll b$, which lead to

$$\exp \left\{-\frac{b + Kc}{f}\right\} \approx 1 - \frac{b + Kc}{f} \approx 1 - \frac{b}{f}.$$

Applying the above approximation to (4.22), we have

$$\begin{aligned} g^{(1)} &\approx \tilde{g}^{(1)}(K) \approx b + Kc + r \left\{1 - \left(1 - \frac{b}{f}\right)^M\right\} \\ &+ \left(\frac{b}{K + 1} + c\right) \left\{1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f}\right\} =: \tilde{\tilde{g}}^{(1)}(K). \end{aligned}$$

It is easy to see that

$$\begin{aligned} \frac{d}{dK} \tilde{\tilde{g}}^{(1)}(K) &= c - \frac{b}{(K + 1)^2} \left\{1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f}\right\}, \\ \frac{d^2}{dK^2} \tilde{\tilde{g}}^{(1)}(K) &= \frac{2b}{(K + 1)^3} \left\{1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f}\right\} \geq 0. \end{aligned}$$

Therefore, $\tilde{\tilde{g}}^{(1)}(K)$ takes its minimum value for K such that

$$\frac{d}{dK} \tilde{\tilde{g}}^{(1)}(K) = c - \frac{b}{(K + 1)^2} \left\{1 - \frac{1 - (1 - b/f)^{M+1}}{(M + 1)b/f}\right\} = 0. \quad (4.23)$$

We now define K_{prop}^* as the number such that $(d/dK)\tilde{g}^{(1)}(K)|_{K=K_{\text{prop}}^*} = 0$, i.e., the solution to (4.23). We then have

$$K_{\text{prop}}^* = \sqrt{\frac{b}{c} \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M+1)b/f} \right\}} - 1.$$

We propose K_{prop}^* as the *optimal number* of checkpoints. Since we have made several approximations in obtaining K_{prop}^* , the proposed number K_{prop}^* is not, in general, equal to the *exact* optimal number of checkpoints, denoted by K^* , which minimizes $g^{(1)}$. Nevertheless, under those approximations, we can expect K_{prop}^* to be a *good* approximation to the *exact* optimal number K^* , i.e.,

$$K^* \approx K_{\text{prop}}^* = \sqrt{\frac{b}{c} \left\{ 1 - \frac{1 - (1 - b/f)^{M+1}}{(M+1)b/f} \right\}} - 1. \quad (4.24)$$

In Subsection 4.5.2, we investigate the accuracy of the approximate formula (4.24) through numerical experiments, where K_{prop}^* is rounded to the nearest integer because the number of checkpoints must be integer.

Finally, we note that the approximation formula (4.24) is independent of r . It is reported by Daly [13] that for single processing, the optimal number of checkpoints is not affected by the time required to resume a failed subtask. Therefore, the formula (4.24) implies that this insensitivity against the time required to resume a failed subtask is also applicable for parallel-distributed processing. Moreover, we also note that if $M = 1$ then

$$K_{\text{prop}}^* = \frac{b}{\sqrt{2cf}} - 1 =: K_{\text{young}}^*, \quad (4.25)$$

where K_{young}^* is the *optimal number* of checkpoints according to Young [49]. Therefore, our result can be considered an generalization of Young's result.

4.5 Numerical Examples

In this section, we show some numerical examples. First, we verify the proposed approximations of the mean task-processing time and the optimal number of checkpoints in comparison with the results of Monte Carlo simulation. In addition, we show the usefulness of the derived optimal number of checkpoints by comparing it with the result of previous study. Finally, we validate through Monte Carlo simulation experiments the assumption of an exponential distribution for the time intervals between consecutive worker failures. Table 4.1 shows the parameter values used in the numerical experiments. We set these parameters according to the literature [6, 15] in order to reflect values which are used in real systems.

Table 4.1: Parameter set.

Parameter	Description	Value
M	Number of workers	10 to 1,000
b	Subtask-processing time	6 to 120 [hour]
c	Time to make a checkpoint	30 to 3,000 [sec]
f	Mean time between worker failures	7 to 180 [day]
r	Time to resume a failed subtask	30 to 3,000 [sec]
K	Number of checkpoints	0 to 30

4.5.1 Approximation Accuracy of the Mean Task-Processing Time

In this subsection, we investigate the approximation accuracy of the mean task-processing time. We apply some approximations to derive the mean task-processing time in Section 4.4. To verify these approximations, we compare the analytical result (4.22) with the result of Monte Carlo simulation. We calculate the 95% confidence interval of the mean task-processing time in simulation experiments.

Figures 4.2 to 4.6 show the mean task-processing time with respect to the number of checkpoints for various system parameters. In Fig. 4.2 (resp. Figs. 4.3, 4.4, 4.5 and 4.6), the parameter M (resp. b , c , f and r) is set to three different values. In these figures, the analytical result gives a good approximation of the mean task-processing time for most parameter values. On the other hand, we observe the gaps between the result of analysis and that of simulation experiment for $M = 1,000$ in Fig. 4.2, $b = 120$ [hour] in Fig. 4.3 and $f = 7$ [day] in Fig. 4.5. These gaps are caused by the ignorance of twice or more worker failures happening to each subtask on analysis in Subsection 4.4.1. However, for such parameter values, qualitative trend against the number of checkpoints is described well by the approximation.

Moreover, in Fig. 4.6, the mean task-processing time takes the minimum value at $K = 13$ for each r . This trend means that the optimal number of checkpoints is not affected by the time required to resume a failed subtask, and agrees with the observation of the approximation formula (4.24).

These results imply that the derived approximation (4.22) is sufficiently accurate to find the number of checkpoints which minimizes the mean task-processing time when the system parameters do not significantly deviate from the approximation that twice or more worker failures on each subtask can be ignored.

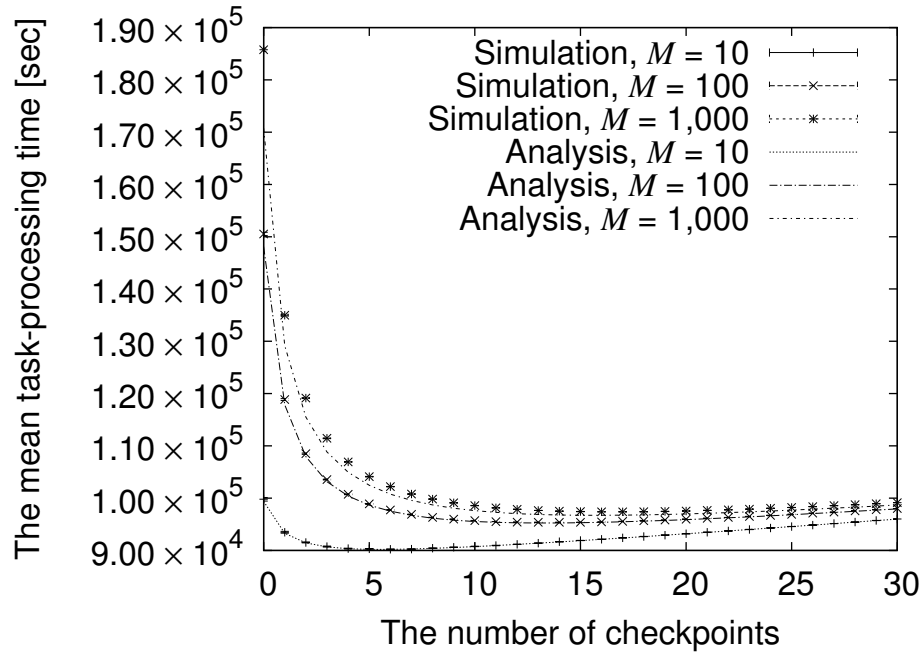


Figure 4.2: Mean task-processing time with respect to the number of checkpoints for various M ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

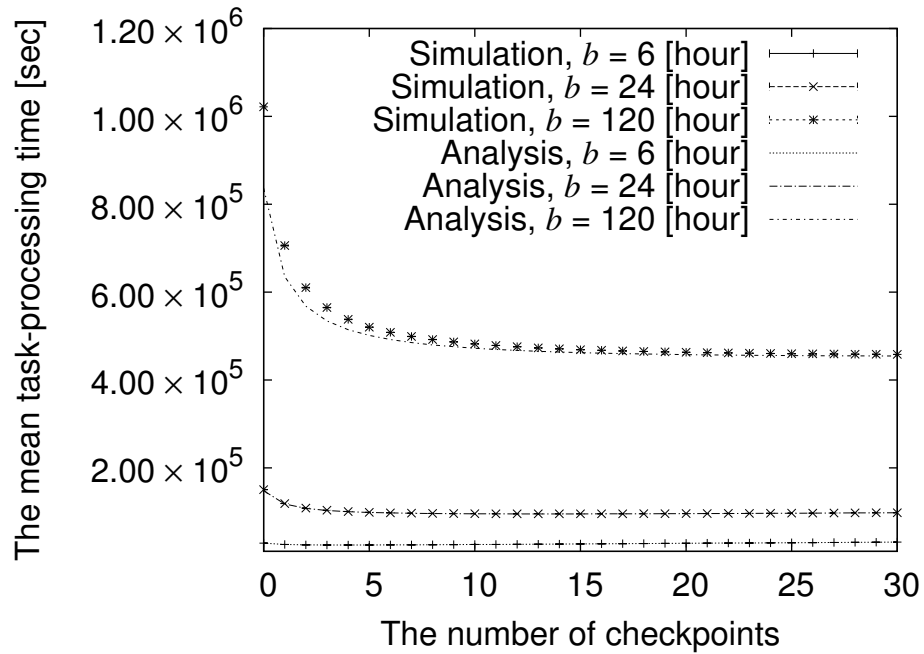


Figure 4.3: Mean task-processing time with respect to the number of checkpoints for various b ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

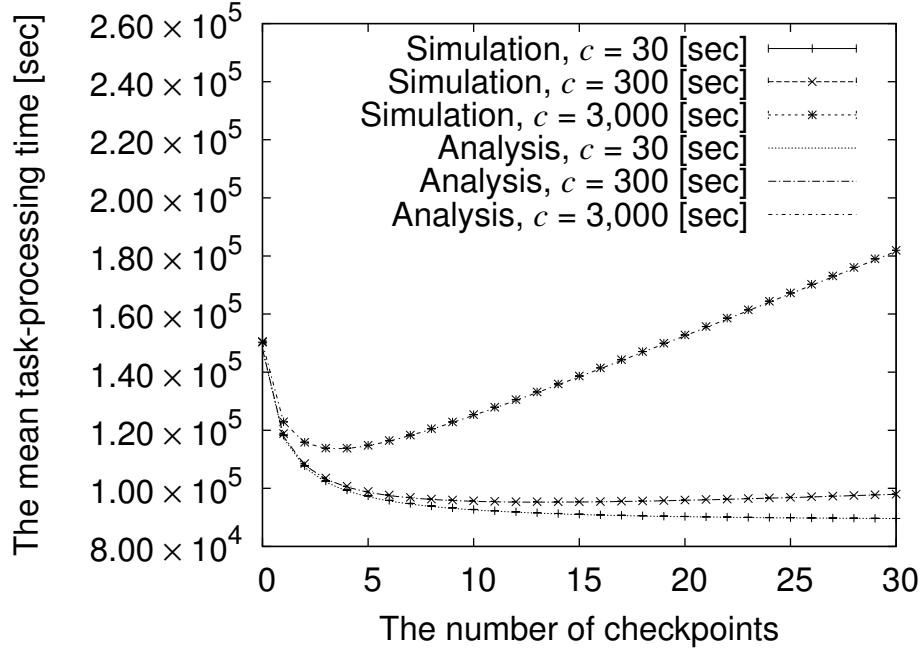


Figure 4.4: Mean task-processing time with respect to the number of checkpoints for various c ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

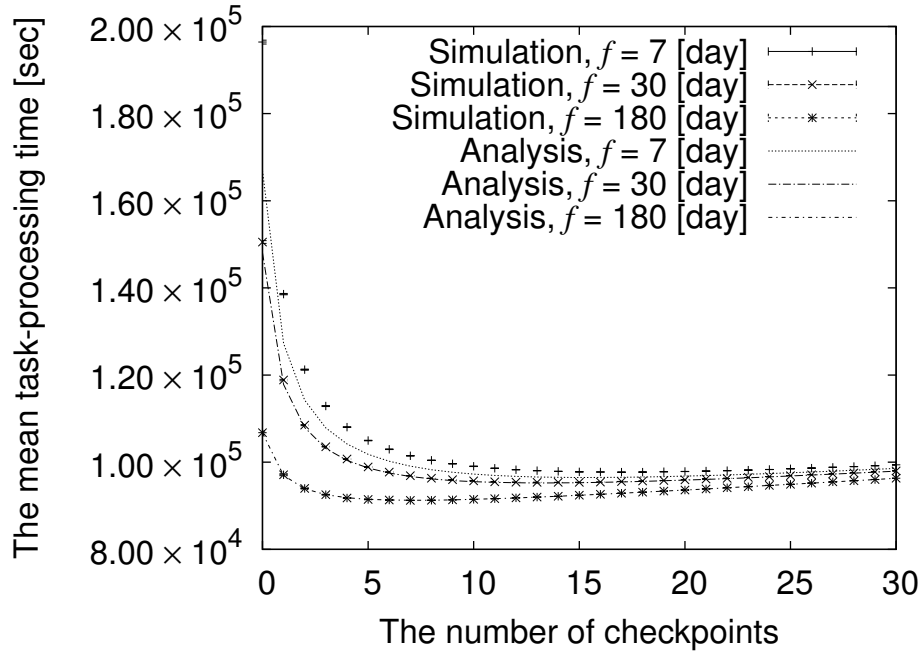


Figure 4.5: Mean task-processing time with respect to the number of checkpoints for various f ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison between the results of analysis and simulation.

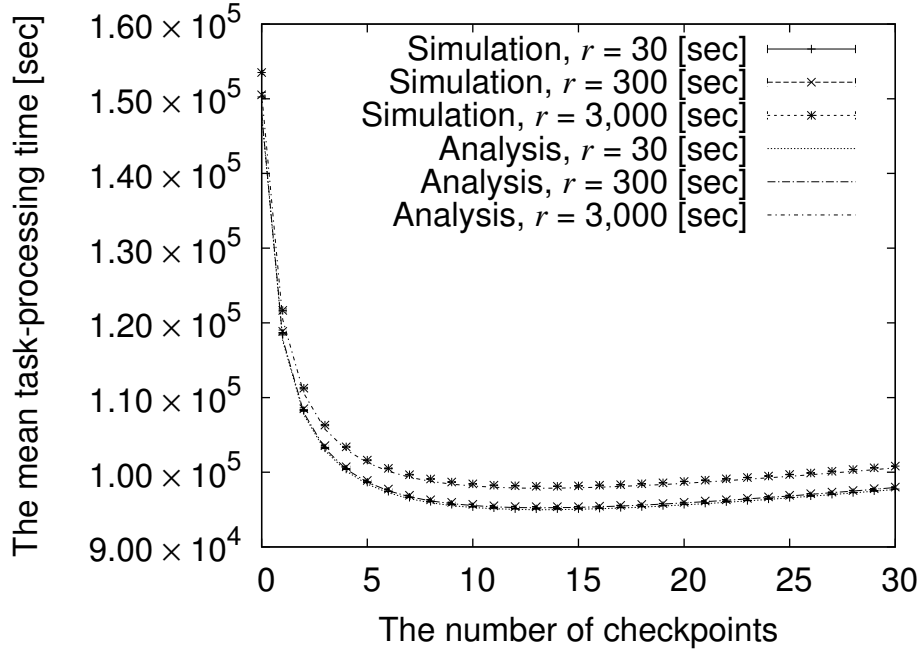


Figure 4.6: Mean task-processing time with respect to the number of checkpoints for various r ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $c = 300$ [sec]): Comparison between the results of analysis and simulation.

4.5.2 Usefulness of the Proposed Approximation for the Optimal Number of Checkpoints

In this subsection, we discuss the usefulness of the proposed approximation for the optimal number of checkpoints. To this end, we compare the mean task-processing times calculated by three different ways: the approximation (4.25) based on Young [49], the proposed approximation (4.24) and Monte Carlo simulation. In simulation experiments, the mean task-processing time is calculated with the 95% confidence interval for each number of checkpoints, and the minimum one is chosen as the result of simulation. Note here that the approximation derived by Young [49] is extended to a higher order one by Daly [13]. However, it is reported in [13] that there is no significant difference between the results of Young and Daly when $b \ll f$ holds. Moreover, as mentioned in Section 4.2, Young's approximation is still applied to the determination of the number of checkpoints for parallel-distributed processing [6, 44]. Therefore, we choose the result of Young as a counterpart.

Figures 4.7 to 4.11 illustrate the mean task-processing times calculated by three different ways. The horizontal axis represents the parameter M (resp. b , c , f and r) in Fig. 4.7 (resp. Figs. 4.8, 4.9, 4.10 and 4.11). Figures 4.7 to 4.11 indicate that the result of the proposed approximation agrees well with that of simulation, and outperforms the result of Young.

This is because Young's approximation does not take into account the issue of stragglers. We can confirm in Figs. 4.7 and 4.9 that this disadvantage appears remarkably for large M and c . These results suggest that the proposed approximation (4.24) for the optimal number of checkpoints is significantly useful to minimize the mean task-processing time.

4.5.3 Assumption Validation of the Analytical Model

In this subsection, we validate the assumption of the proposed analytical model. We assume in our analytical model that the time intervals between consecutive worker failures follow an exponential distribution to simplify the analysis. However, there are various discussions on its validity, and some studies point out that an exponential distribution is poorly fitted to the distribution for the time intervals between consecutive worker failures [6, 18, 30]. Therefore, we calculate the mean task-processing time through Monte Carlo simulation when the time intervals between consecutive worker failures follow an exponential distribution, gamma distribution with shape parameter 0.35 or Weibull distribution with shape parameter 0.48. We choose the latter two distributions according to the measurement of real systems [30]. The mean task-processing time is calculated with the 95% confidence interval for each number of checkpoints, and the minimum one is chosen as the result for each distribution.

Figures 4.12 to 4.16 show the mean task-processing time when the time intervals between consecutive worker failures follow the exponential, gamma or Weibull distribution. The horizontal axis represents the parameter M (resp. b , c , f and r) in Fig. 4.12 (resp. Figs. 4.13, 4.14, 4.15 and 4.16). In these figures, there is no significant difference among the mean task-processing time for three distributions, and we can consider that the assumption that the time intervals between consecutive worker failures follow an exponential distribution is reasonable.

To consider this insensitivity against the shape of the distribution, we illustrate the mean task-processing time with respect to small f in the cases of three distributions in Fig. 4.17. In this figure, the difference between the result of the exponential distribution and that of the gamma or Weibull distribution decreases with the increase in the mean time between worker failures. This implies that the shape of the distribution does not significantly affect the mean task-processing time when the mean time between worker failures is sufficiently larger than the subtask processing time (i.e. $b \ll f$).

These results indicate that the proposed analytical model can evaluate the mean task-processing time for parallel-distributed processing with checkpointing method even when the time intervals between consecutive worker failures follow a more realistic distribution.

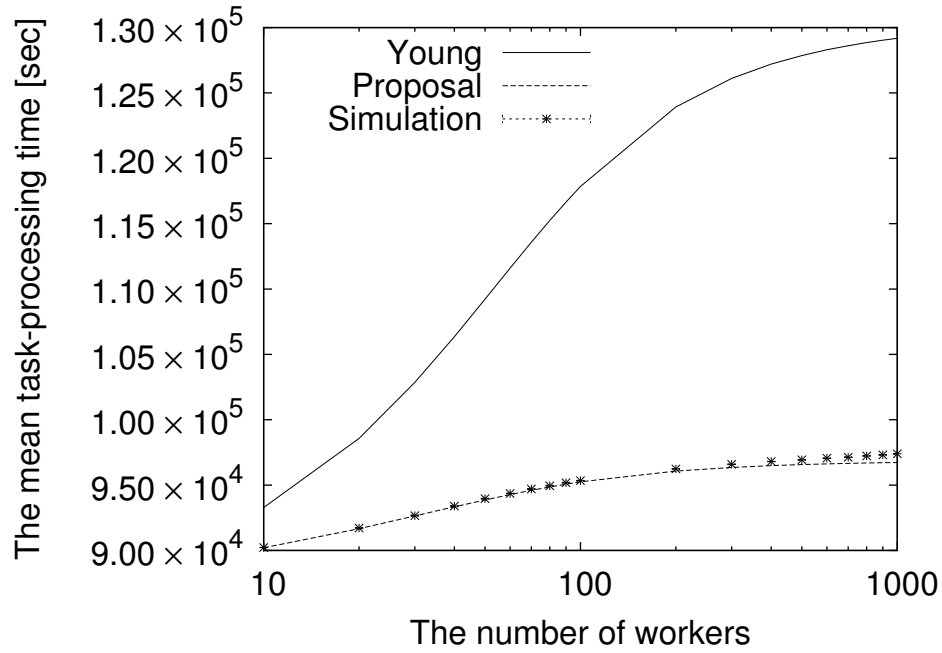


Figure 4.7: Mean task-processing time with respect to M for the optimal number of checkpoints ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.

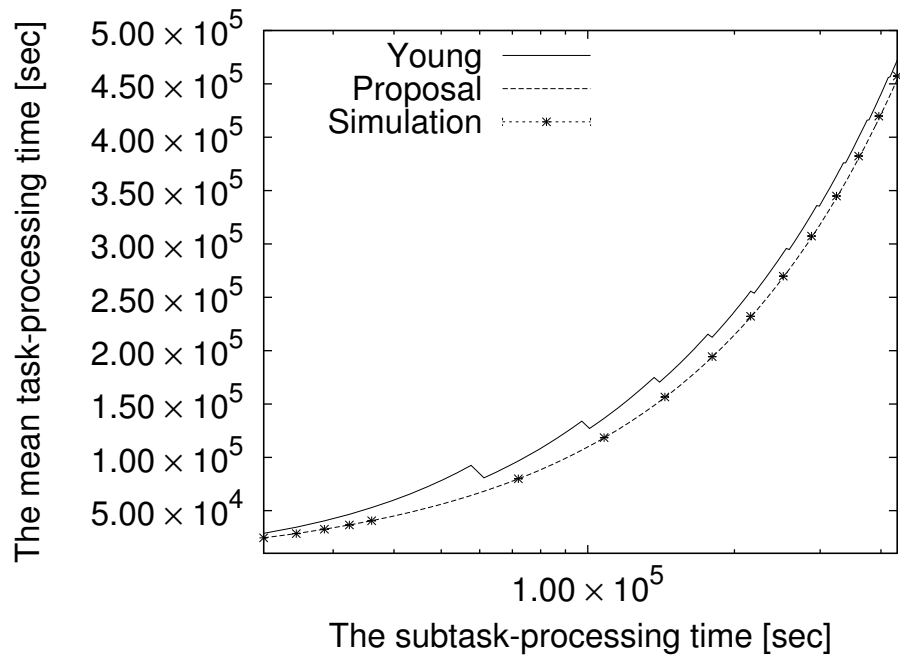


Figure 4.8: Mean task-processing time with respect to b for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.

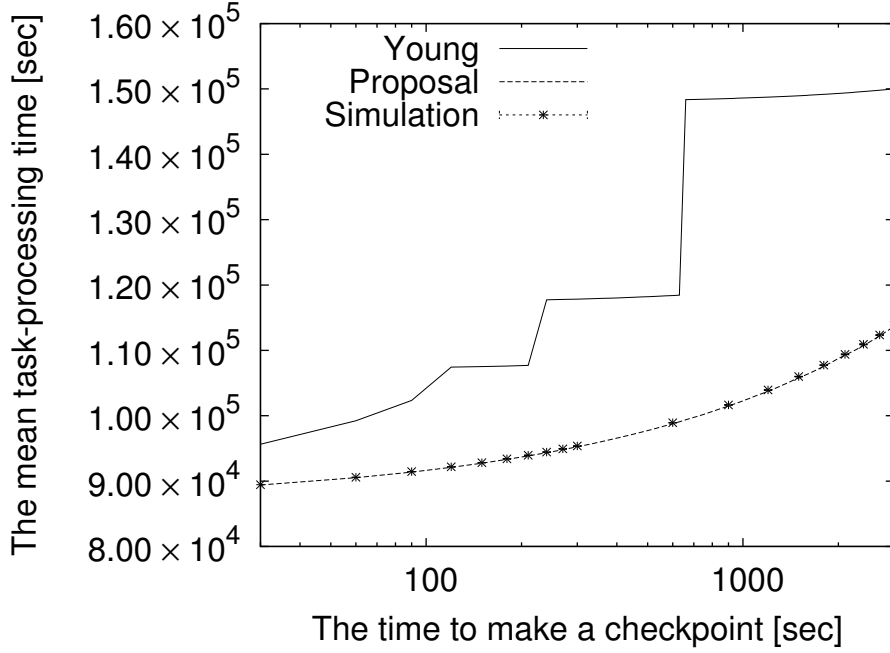


Figure 4.9: Mean task-processing time with respect to c for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.

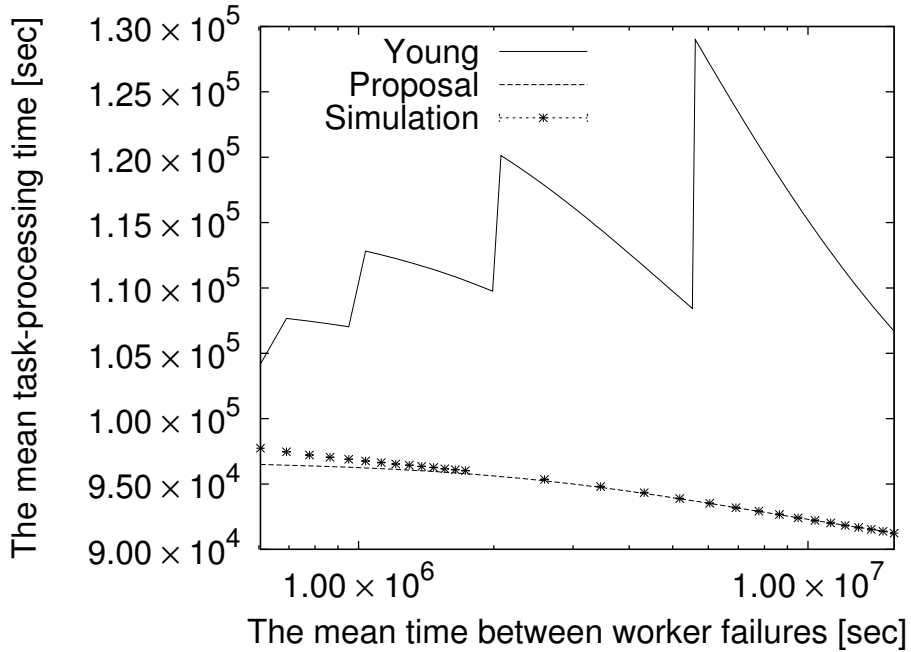


Figure 4.10: Mean task-processing time with respect to f for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison between the results of previous and proposal analyses and simulation.

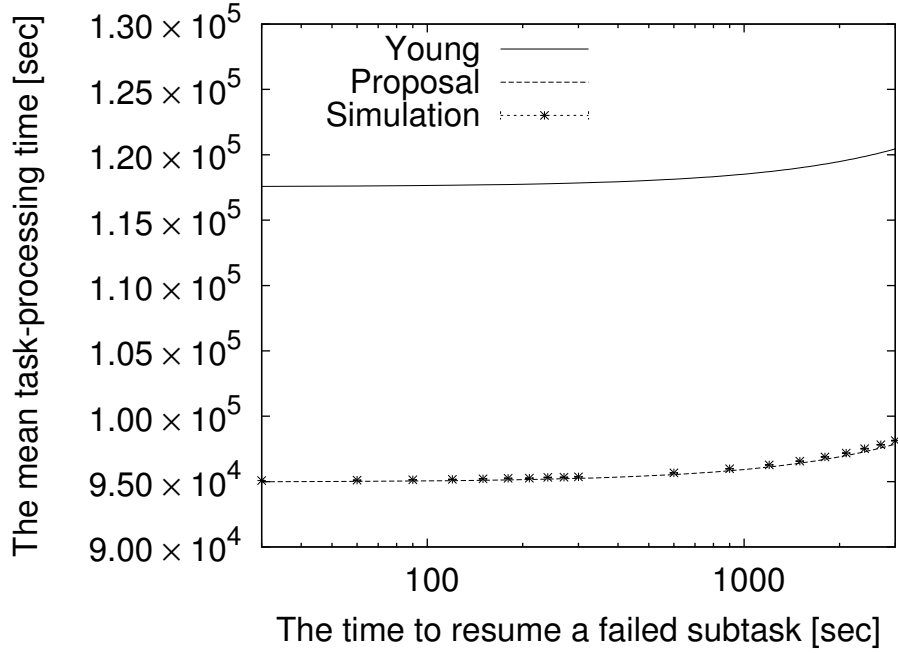


Figure 4.11: Mean task-processing time with respect to r for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $b = 24$ [hour], $f = 30$ [day]): Comparison between the results of previous and proposal analyses and simulation.

4.6 Summary

In this chapter, we evaluated the effect of checkpointing method on the task-processing time and considered the optimal number of checkpoints which achieves the shortest task-processing time. We constructed an analytical model for large-scale parallel-distributed processing with checkpointing, and approximately derived explicit expressions for the mean task-processing time and the optimal number of checkpoints. In numerical examples, we confirmed the accuracy of the derived approximations in comparison with the results of Monte Carlo simulation. Moreover, we showed the usefulness of the derived optimal number of checkpoints by comparing it with the result of previous study. Finally, we validated through Monte Carlo simulation experiments the assumption of the analytical model that the time intervals between consecutive worker failures follow an exponential distribution. We can claim that the proposed approximations are sufficiently accurate even when the time intervals between consecutive worker failures follow a more realistic distribution, such as a gamma or Weibull distribution. Furthermore, the derived optimal number of checkpoints is as simple as Young's formula [49], while our formula outperforms Young's one and is useful for minimizing the task-processing time on parallel-distributed processing.

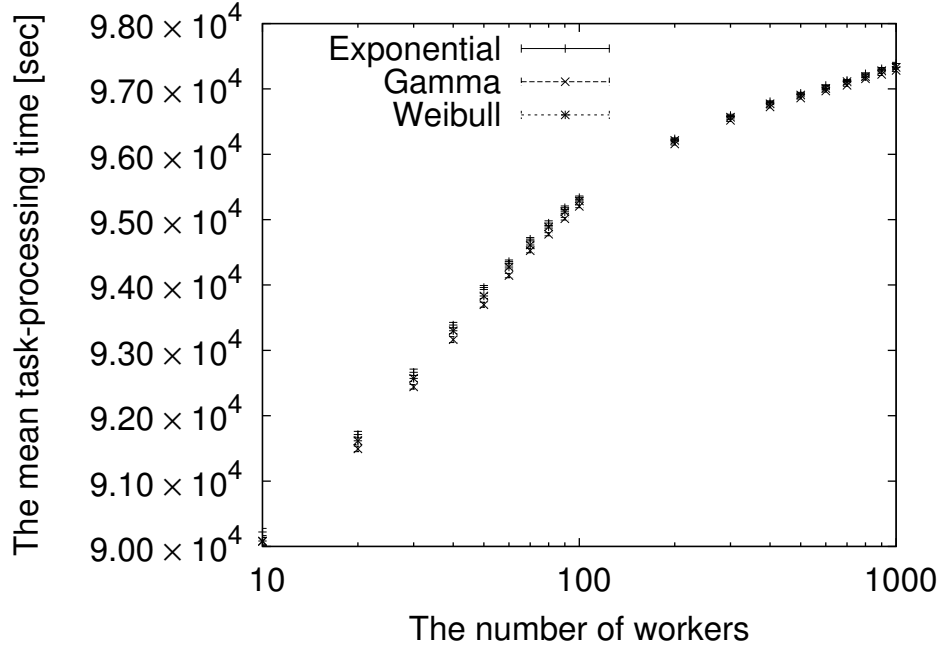


Figure 4.12: Mean task-processing time with respect to M for the optimal number of checkpoints ($b = 24$ [hour], $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

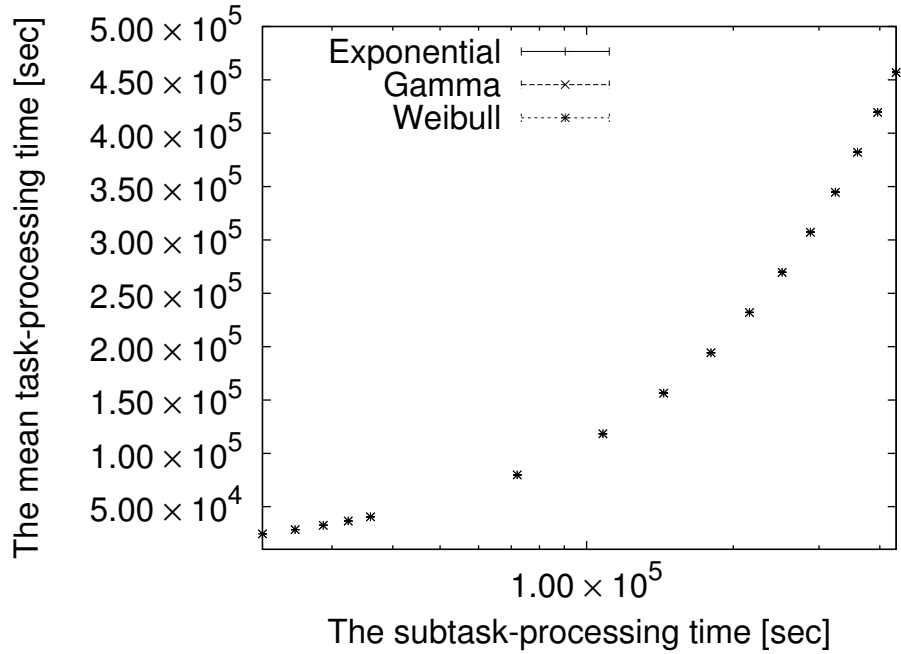


Figure 4.13: Mean task-processing time with respect to b for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

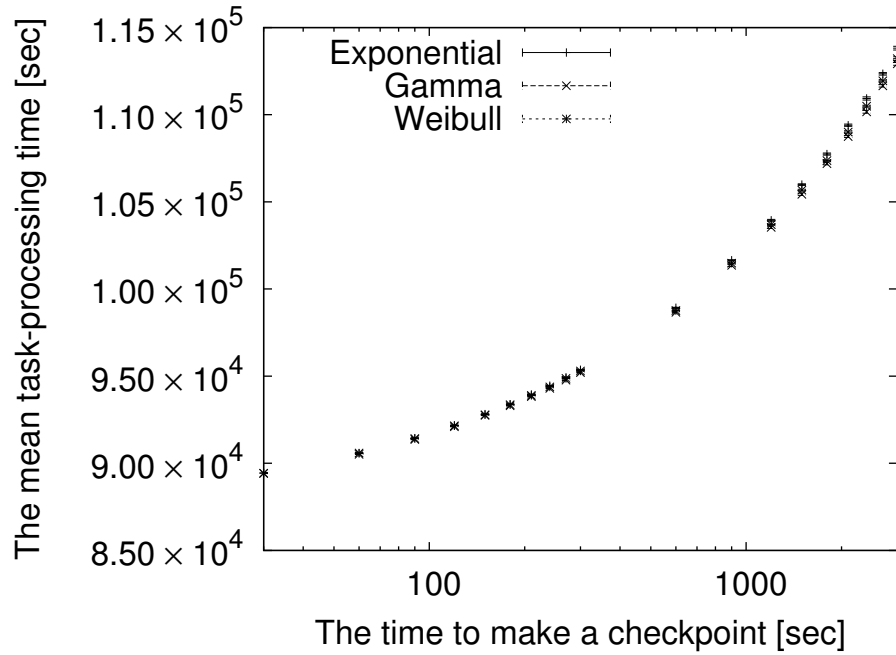


Figure 4.14: Mean task-processing time with respect to c for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $f = 30$ [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

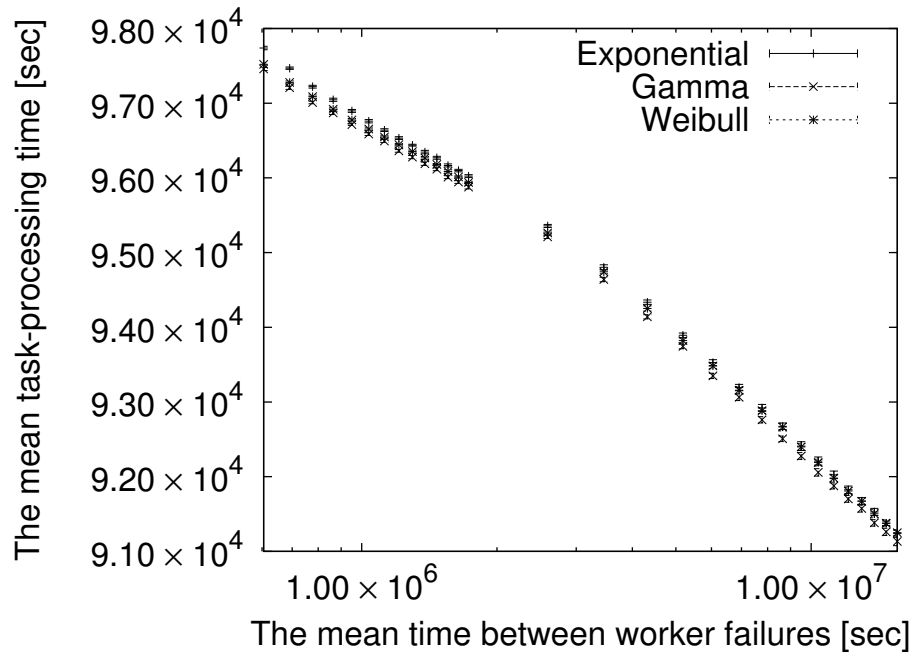


Figure 4.15: Mean task-processing time with respect to f for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

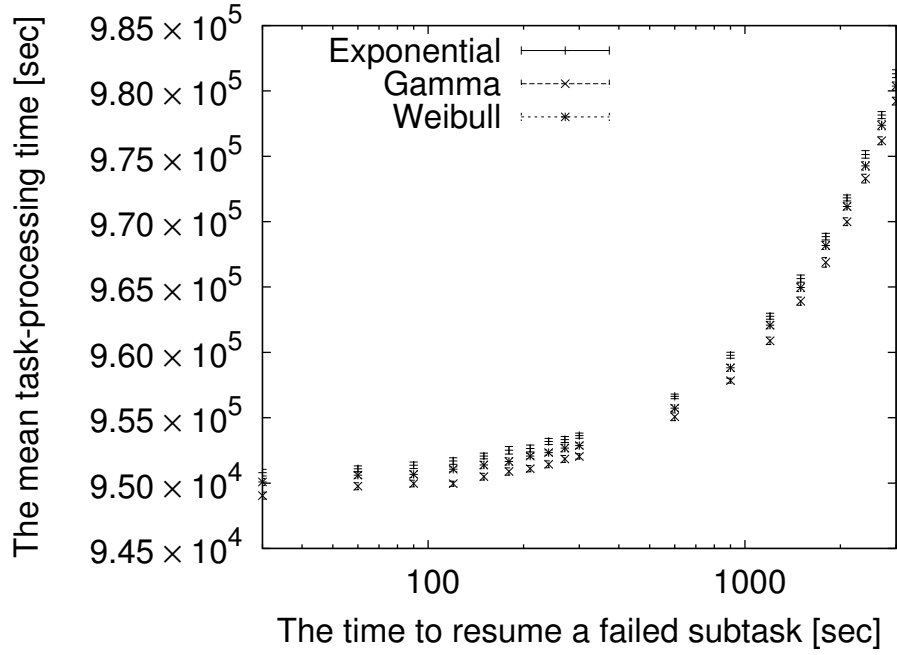


Figure 4.16: Mean task-processing time with respect to r for the optimal number of checkpoints ($M = 100$, $c = 300$ [sec], $b = 24$ [hour], $f = 30$ [day]): Comparison among three distributions for the time intervals between consecutive worker failures.

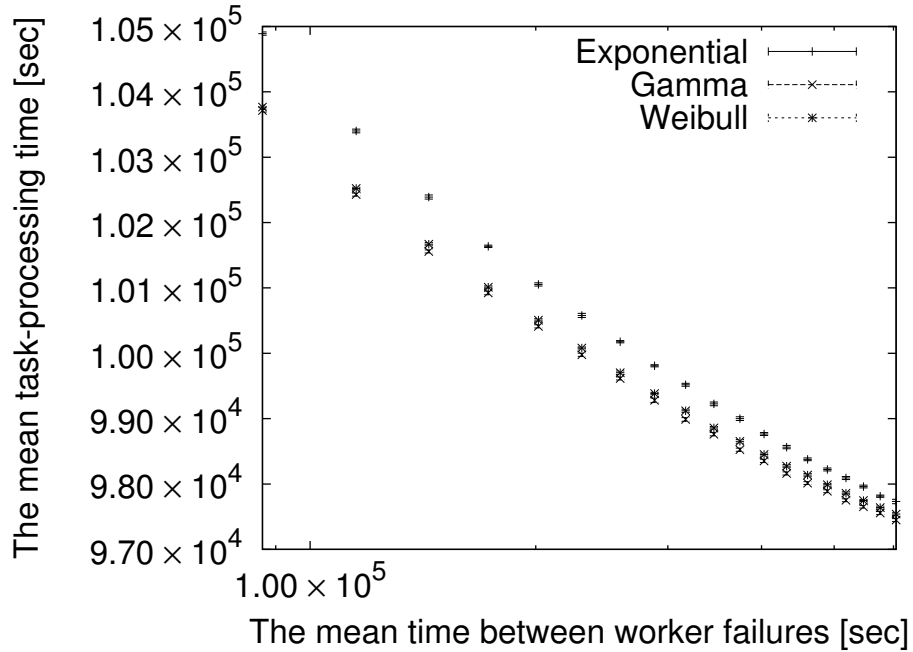


Figure 4.17: Mean task-processing time with respect to small f for the optimal number of checkpoints ($M = 100$, $b = 24$ [hour], $c = 300$ [sec], $f = 1$ to 7 [day], $r = 300$ [sec]): Comparison among three distributions for the time intervals between consecutive worker failures.

Chapter 5

Conclusion

This dissertation considered the impact of stragglers and the effect of task replication and checkpointing on the performance of parallel-distributed processing through analytical models and their evaluations.

In Chapter 2, we considered the impact of stragglers and the efficiency of backup-task scheduling. We modeled the task-scheduling server as a single-server queue with many workers, deriving the maximum throughput, mean response time and mean total processing time. From the numerical results, we confirmed that the issue of stragglers degrades the performance when the coefficient of variation of the worker-processing time is large. Moreover, we showed that backup-task scheduling is significantly efficient for improving the performance in the case of a large variance of the worker-processing time.

Chapter 3 evaluated the efficiency of task-replication scheduling in terms of the number of replications. To this end, the mean value and standard deviation of the task-processing time were approximately derived with extreme value theory, whereas the mean total processing time was exactly evaluated. Through numerical analysis, we showed that the efficiency of task-replication scheduling is improved significantly when the coefficient of variation of the worker-processing time increases. However, this efficiency depends significantly on the tail of the worker-processing time distribution even when the means and variances of the distributions are the same. In addition, we also showed that the optimal number of replications which achieves the shortest task-processing time mainly depends on the coefficient of variation of the worker-processing time.

In Chapter 4, we evaluated the effect of checkpointing method on the task-processing time and considered the optimal number of checkpoints which achieves the shortest task-processing time. We constructed an analytical model for parallel-distributed processing with checkpointing, and approximately derived explicit expressions for the mean task-processing time and the optimal number of checkpoints. In numerical examples, we showed the usefulness of the

derived optimal number of checkpoints by comparing it with the result of previous study.

Finally, we suggest some directions for future research. In Chapter 3, numerical examples showed that the proposed analytical model is not useful for evaluating a kind of task-replication scheduling in which backup executions are activated when the elapsed time of the processing of a subtask is greater than a pre-specified threshold. This type of task-replication scheduling is implemented in a real system, and we need further development of the proposed analytical model. In order to treat this case, we can readily consider an extended model from the one in Chapter 3. However, the way to analyze the extended model is not trivial because the distribution function of the subtask-processing time is described as a more complicated formula than that in Chapter 3. Hashimoto et al. [25] analyze this extended model when the number of replications is one, and their derivation could be applied in the case of an arbitrary number of replications.

Furthermore, we can consider parallel-distributed processing with both task replication and checkpointing. In this processing, the progress of the processing is periodically saved as a checkpoint during a period of the processing of a subtask, and replications of the subtask begin to be processed from the latest checkpoint when the elapsed time of the processing of the original subtask exceeds a pre-specified threshold. The optimal configuration of parameters for this processing might be different from that for processing with either task replication or checkpointing, and this processing could reduce the redundant consumption of computing resources. Therefore, it is useful to construct and evaluate analytical models for parallel-distributed processing with both methods. We could also treat this case by extending the study in Chapter 3.

Appendix A

Preliminary Analysis Results

This chapter summarizes the preliminary results on basic extreme value theory, which are used in Subsection 3.4.2.

Let $\{X, X_k; k = 1, 2, \dots, n\}$ denote a sequence of independent and identically distributed random variables with a distribution function F , which is non-degenerate, i.e., $F(x_F) = 1$, where $x_F = \sup\{x \in \mathbb{R} := (-\infty, \infty); F(x) < 1\}$.

Let $\bar{X}_n = \max_{1 \leq k \leq n} X_k$ for $n = 1, 2, \dots$. It follows from the fundamental Fisher-Tippett theorem (see, e.g., Theorem 3.2.3 in [21]) that if there exist some $c_n > 0$ and $d_n \in \mathbb{R}$ ($n = 1, 2, \dots$) such that the distribution of $(\bar{X}_n - d_n)/c_n$ weakly converges to a non-degenerate distribution Θ , i.e.,

$$\lim_{n \rightarrow \infty} P\left(\frac{\bar{X}_n - d_n}{c_n} \leq x\right) = \Theta(x), \quad (\text{A.1})$$

for any $x \in \mathbb{R}$ such that Θ is continuous, then Θ must be one of the following three standard *extreme value distributions*:

$$\begin{aligned} \text{Fréchet :} \quad & \Phi_\alpha(x) = \begin{cases} 0, & x \leq 0, \\ \exp\{-x^{-\alpha}\}, & x > 0, \end{cases} & \alpha > 0, \\ \text{Weibull :} \quad & \Psi_\alpha(x) = \begin{cases} \exp\{-(-x)^\alpha\}, & x \leq 0, \\ 1, & x > 0, \end{cases} & \alpha > 0, \\ \text{Gumbel :} \quad & \Lambda(x) = \exp\{-\exp\{-x\}\}, & x \in \mathbb{R}. \end{aligned}$$

For simplicity, according to [21], we introduce the following notation.

Definition A.0.1 The random variable X and its distribution F are said to be in the maximum domain of attraction of the extreme value distribution Θ (denoted by $X \in \text{MDA}(\Theta)$ and $F \in \text{MDA}(\Theta)$) if there exist some $c_n > 0$ and $d_n \in \mathbb{R}$ ($n = 1, 2, \dots$) such that (A.1) holds.

In Subsection 3.4.2, we use asymptotic results associated with the two classes $\text{MDA}(\Lambda)$ and $\text{MDA}(\Phi_\alpha)$, which are described in Subsections A.1 and A.2.

A.1 Maximum Domain of Attraction of a Gumbel Distribution

Proposition A.1.1 (Theorem 3.3.26 in [21]) $F \in \text{MDA}(\Lambda)$ if and only if there exists some $x_0 < x_F$ such that F has the following representation:

$$1 - F(x) = c(x) \exp \left\{ - \int_{x_0}^x \frac{g(t)}{a(t)} dt \right\}, \quad x_0 < x < x_F, \quad (\text{A.2})$$

where c and g are measurable functions such that $\lim_{x \uparrow x_F} c(x) = c > 0$ and $\lim_{x \uparrow x_F} g(x) = 1$; and where $a(\cdot) > 0$ is an absolutely continuous function with respect to the Lebesgue measure and its density $a'(\cdot)$ satisfies $\lim_{x \uparrow x_F} a'(x) = 0$. In addition, we can choose the normalizing constants c_n and d_n in (A.1) as follows:

$$c_n = a(d_n), \quad d_n = F^{-1} \left(1 - \frac{1}{n} \right), \quad (\text{A.3})$$

where $F^{-1}(x) = \inf\{y; F(y) \geq x\}$.

Proposition A.1.2 (Proposition 2.1 (iii) in [41]) If $F \in \text{MDA}(\Lambda)$ and

$$\int_{-\infty}^0 |x|^k dF(x) < \infty,$$

for some integer $k > 0$, then

$$\lim_{n \rightarrow \infty} \mathbb{E} \left[\left(\frac{\bar{X}_n - d_n}{c_n} \right)^k \right] = (-1)^k \lim_{x \rightarrow 1} \frac{d^k}{dx^k} \Gamma(x),$$

where Γ denotes the Gamma function and c_n and d_n are given by (A.3).

A.2 Maximum Domain of Attraction of a Fréchet Distribution

Proposition A.2.1 (Theorem 3.3.7 in [21]) $F \in \text{MDA}(\Phi_\alpha)$ if and only if the tail distribution $1 - F$ is regularly varying with index $-\alpha$, i.e.,

$$\lim_{x \rightarrow \infty} \frac{1 - F(xt)}{1 - F(x)} = t^{-\alpha}, \quad t > 0. \quad (\text{A.4})$$

The normalizing constants c_n and d_n can be chosen as

$$c_n = F^{-1} \left(1 - \frac{1}{n} \right), \quad d_n = 0. \quad (\text{A.5})$$

Proposition A.2.2 (Proposition 2.1 (i) in [41]) If $F \in \text{MDA}(\Phi_\alpha)$ and

$$\int_{-\infty}^0 |x|^k dF(x) < \infty,$$

for some integer $0 < k < \alpha$, then

$$\lim_{n \rightarrow \infty} \mathbb{E} \left[\left(\frac{\bar{X}_n}{c_n} \right)^k \right] = \Gamma \left(1 - \frac{k}{\alpha} \right),$$

where c_n is given by (A.5).

Bibliography

- [1] S. Ali, B. Eslamnour and Z. Shah, “A Case for On-Machine Load Balancing,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 556–564, 2011.
- [2] G. Ananthanarayanan, A. Ghodsi, A. Wang, D. Borthakur, S. Kandula, S. Shenker and I. Stoica, “PACMan: Coordinated Memory Caching for Parallel Jobs,” in *Proc. the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI 2012)*, 2012, pp. 20–20.
- [3] C. Anglano and M. Canonico, “Scheduling Algorithms for Multiple Bag-of-Task Applications on Desktop Grids: A Knowledge-Free Approach,” in *Proc. the IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, 2008, pp. 1–8.
- [4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, “A View of Cloud Computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [5] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool, 2009.
- [6] T. C. Bressoud and M. A. Kozuch, “Cluster Fault-Tolerance: An Experimental Evaluation of Checkpointing and MapReduce through Simulation,” in *Proc. the IEEE International Conference on Cluster Computing and Workshops (CLUSTER 2009)*, 2009.
- [7] C. L. P. Chen and C.-Y. Zhang, “Data-Intensive Applications, Challenges, Techniques and Technologies: A Survey on Big Data,” *Information Sciences*, vol. 275, pp. 314–347, 2014.
- [8] M. Chen, S. Mao and Y. Liu, “Big Data: A Survey,” *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.
- [9] Y. Chen, A. Ganapathi, R. Griffith and R. Katz, “The Case for Evaluating MapReduce Performance Using Workload Suites,” in *Proc. the 19th IEEE International Symposium*

- on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2011)*, 2011, pp. 390–399.
- [10] Y. Chen, S. Alspaugh and R. Katz, “Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads,” *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.
- [11] W. Cirne, D. Paranhos, F. Brasileiro and L. F. W. Góes, “On the Efficacy, Efficiency and Emergent Behavior of Task Replication in Large Distributed Systems,” *Parallel Computing*, vol. 33, no. 3, pp. 213–234, 2007.
- [12] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy and R. Sears, “MapReduce Online,” in *Proc. the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2010)*, 2010.
- [13] J. T. Daly, “A Higher Order Estimate of the Optimum Checkpoint Interval for Restart Dumps,” *Future Generation Computer Systems*, vol. 22, no. 3, pp. 303–312, 2006.
- [14] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [15] J. Dean, “Designs, Lessons and Advice from Building Large Distributed Systems,” in *Keynote Presentation of the 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS 2009)*, 2009.
- [16] J. Dean and S. Ghemawat, “MapReduce: A Flexible Data Processing Tool,” *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.
- [17] J. Dejun, G. Pierre and C.-H. Chi, “EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications,” in *Proc. the 7th International Conference on Service-Oriented Computing (ICSOC 2009)*, 2009, pp. 197–207.
- [18] S. Di, Y. Robert, F. Vivien, D. Kondo, C.-L. Wang and F. Cappello, “Optimization of Cloud Task Processing with Checkpoint-Restart Mechanism,” in *Proc. the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2013)*, 2013.
- [19] T. Dillon, C. Wu and E. Chang, “Cloud Computing: Issues and Challenges,” in *Proc. the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010)*, 2010, pp. 27–33.

- [20] M. Dobber, R. V. D. Mei and G. Koole, “Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 2, pp. 207–218, 2009.
- [21] P. Embrechts, C. Klüppelberg and T. Mikosch, *Modelling Extremal Events for Insurance and Finance*. Springer, 1997.
- [22] L. Fialho, D. Rexachs and E. Luque, “What is Missing in Current Checkpoint Interval Models?,” in *Proc. the 31st International Conference on Distributed Computing Systems (ICDCS 2011)*, 2011, pp. 322–332.
- [23] C. P. Gomes, B. Selman, N. Crato and H. Kautz, “Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems,” *Journal of Automated Reasoning*, vol. 24, no. 1–2, pp. 67–100, 2000.
- [24] D. Gross, J. F. Shortle, J. M. Thompson and C. M. Harris, *Fundamentals of Queueing Theory*, 4th ed. John Wiley & Sons, 2008.
- [25] K. Hashimoto, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Analysis of Backup-Task Scheduling with Deadline Time in Cloud Computing,” *Journal of Industrial and Management Optimization*, vol. 11, no. 3, pp. 867–886, 2015.
- [26] T. Hirai, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Analysis of Large-Scale Parallel-Distributed Processing with Backup Tasks for Cloud Computing,” *Journal of Industrial and Management Optimization*, vol. 10, no. 1, pp. 113–129, 2014.
- [27] T. Hirai, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Analysis of Task Replication in Large-Scale Parallel-Distributed Processing: An Extreme Value Theory Approach,” *Journal of the Operations Research Society of Japan*, vol. 59, no. 2, pp. 174–194, 2016.
- [28] T. Hirai, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Optimization of Parallel-Distributed Processing with Checkpointing for Cloud Environment,” Published online in *Journal of Industrial and Management Optimization*.
- [29] B. Javadi, D. Kondo, J.-M. Vincent and D. P. Anderson, “Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1896–1903, 2011.
- [30] B. Javadi, D. Kondo, A. Iosup and D. Epema, “The Failure Trace Archive: Enabling the Comparison of Failure Measurements and Models of Distributed Systems,” *Journal of Parallel and Distributed Computing*, vol. 73, no. 8, pp. 1208–1223, 2013.

- [31] H. Jin, Y. Chen, H. Zhu and X.-H. Sun, “Optimizing HPC Fault-Tolerant Environment: An Analytical Approach,” in *Proc. the 39th International Conference on Parallel Processing (ICPP 2010)*, 2010, pp. 525–534.
- [32] V. Kalavri and V. Vlassov, “MapReduce: Limitations, Optimizations and Open Issues,” in *Proc. the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM 2013)*, 2013, pp. 1031–1038.
- [33] D. Kondo, B. Javadi, A. Iosup and D. Epema, “The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems,” in *Proc. the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID 2010)*, 2010, pp. 398–407.
- [34] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung and B. Moon, “Parallel Data Processing with MapReduce: A Survey,” *ACM SIGMOD Record*, vol. 40, no. 4, pp. 11–20, 2011.
- [35] H. Lin, X. Ma, J. Archuleta, W. Feng, M. Gardner and Z. Zhang, “MOON: MapReduce on Opportunistic Environments,” in *Proc. the 19th ACM International Symposium on High Performance Distributed Computing (HPDC 2010)*, 2010, pp. 95–106.
- [36] A. Martin, T. Knauth, S. Creutz, D. Becker, S. Weigert, C. Fetzer and A. Brito, “Low-Overhead Fault Tolerance for High-Throughput Data Processing Systems,” in *Proc. the 31st International Conference on Distributed Computing Systems (ICDCS 2011)*, 2011, pp. 689–699.
- [37] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” *Recommendations of the National Institute of Standards and Technology*, NIST Special Publication 800–145, 2011.
- [38] N. Nóbrega-Júnior, L. Assis and F. Brasileiro, “Scheduling CPU-Intensive Grid Applications Using Partial Information,” in *Proc. the 37th International Conference on Parallel Processing (ICPP 2008)*, 2008, pp. 262–269.
- [39] J. O’Loughlin and L. Gillam, “Towards Performance Prediction for Public Infrastructure Clouds: An EC2 Case Study,” in *Proc. the 5th IEEE International Conference on Cloud Computing Technology and Science (CLOUDCOM 2013)*, 2013, pp. 475–480.
- [40] F. W. J. Olver, D. W. Lozier, R. F. Boisvert and C. W. Clark, *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.
- [41] S. I. Resnick, *Extreme Values, Regular Variation and Point Processes*. Springer, 2008.

- [42] S. Sakr, A. Liu, D. M. Batista and M. Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 311–336, 2011.
- [43] M. Silberstein, "Building an Online Domain-Specific Computing Service over Non-Dedicated Grid and Cloud Resources: The Superlink-Online Experience," in *Proc. the 11th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CC-GRID 2011)*, 2011, pp. 174–183.
- [44] M. Taifi, J. Y. Shi and A. Khreishah, "SpotMPI: A Framework for Auction-Based HPC Computing Using Amazon Spot Instances," in *Proc. the 11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2011)*, 2011, pp. 109–120.
- [45] T. White, *Hadoop: The Definitive Guide*, 2nd ed. O'Reilly Media, 2010.
- [46] K. Wolter, "Stochastic Models for Restart, Rejuvenation and Checkpointing," *Habilitation Thesis*, Humboldt University of Berlin, 2008.
- [47] K. Xiong and H. Perros, "Service Performance and Analysis in Cloud Computing," in *Proc. the IEEE Congress on Services (SERVICES 2009)*, 2009, pp. 693–700.
- [48] N. Yigitbasi, A. Iosup, D. Epema and S. Ostermann, "C-Meter: A Framework for Performance Analysis of Computing Clouds," in *Proc. the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2009)*, 2009, pp. 472–477.
- [49] J. W. Young, "A First Order Approximation to the Optimum Checkpoint Interval," *Communications of the ACM*, vol. 17, no. 9, pp. 530–531, 1974.
- [50] L. Youseff, M. Butrico and D. D. Silva, "Toward a Unified Ontology of Cloud Computing," in *Proc. the Grid Computing Environments Workshop (GCE 2008)*, 2008.
- [51] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," in *Proc. the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2008)*, 2008, pp. 29–42.
- [52] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker and I. Stoica, "Discretized Streams: Fault-Tolerant Streaming Computation at Scale," in *Proc. the 24th ACM Symposium on Operating Systems Principles (SOSP 2013)*, 2013, pp. 423–438.
- [53] Q. Zhang, L. Cheng and R. Boutaba, "Cloud Computing: State-of-the-Art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

Copyright Notice

Chapter 2 © 2014 the American Institute of Mathematical Sciences. Re-used with permission from T. Hirai, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Analysis of Large-Scale Parallel-Distributed Processing with Backup Tasks for Cloud Computing,” *Journal of Industrial and Management Optimization*, vol. 10, no. 1, pp. 113–129, 2014 (DOI: 10.3934/jimo.2014.10.113).

Chapter 3 and Appendix A © 2016 the Operations Research Society of Japan. Re-used with permission from T. Hirai, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Analysis of Task Replication in Large-Scale Parallel-Distributed Processing: An Extreme Value Theory Approach,” *Journal of the Operations Research Society of Japan*, vol. 59, no. 2, pp. 174–194, 2016 (DOI: 10.15807/jorsj.59.174).

Chapter 4 © 2018 the American Institute of Mathematical Sciences. Re-used with permission from T. Hirai, H. Masuyama, S. Kasahara and Y. Takahashi, “Performance Optimization of Parallel-Distributed Processing with Checkpointing for Cloud Environment,” Published online in *Journal of Industrial and Management Optimization* (DOI: 10.3934/jimo.2018014).