University of Central Florida
STARS

Electronic Theses and Dissertations, 2004-2019

2016

Developing New Power Management and High-Reliability Schemes in Data-Intensive Environment

Ruijun Wang
University of Central Florida

Part of the Computer Engineering Commons
Find similar works at: https://stars.library.ucf.edu/etd
University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
Wang, Ruijun, "Developing New Power Management and High-Reliability Schemes in Data-Intensive Environment" (2016). *Electronic Theses and Dissertations, 2004-2019.* 5433.
https://stars.library.ucf.edu/etd/5433

Showcase of Text, Archives, Research & Scholarship

DEVELOPING NEW POWER MANAGEMENT AND HIGH-RELIABILITY SCHEMES IN DATA-INTENSIVE ENVIRONMENT

by

RUIJUN WANG
M.S.Information Systems, Central Queensland University, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy of Computer Engineering
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2016

Major Professor: Jun Wang

© 2016 Ruijun Wang

ABSTRACT

With the increasing popularity of data-intensive applications as well as the large-scale computing and storage systems, current data centers and supercomputers are often dealing with extremely large data-sets. To store and process this huge amount of data reliably and energy-efficiently, three major challenges should be taken into consideration for the system designers. Firstly, power conservation–Multicore processors or CMPs have become a mainstream in the current processor market because of the tremendous improvement in transistor density and the advancement in semiconductor technology. However, the increasing number of transistors on a single die or chip reveals a super-linear growth in power consumption [4]. Thus, how to balance system performance and power-saving is a critical issue which needs to be solved effectively. Secondly, system reliability–Reliability is a critical metric in the design and development of replication-based big data storage systems such as Hadoop File System (HDFS). In the system with thousands machines and storage devices, even in-frequent failures become likely. In Google File System, the annual disk failure rate is 2.88%,which means you were expected to see 8,760 disk failures in a year. Unfortunately, given an increasing number of node failures, how often a cluster starts losing data when being scaled out is not well investigated. Thirdly, energy efficiency–The fast processing speeds of the current generation of supercomputers provide a great convenience to scientists dealing with extremely large data sets. The next generation of ”exascale” supercomputers could provide accurate simulation results for the automobile industry, aerospace industry, and even nuclear fusion reactors for the very first time. However, the energy cost of super-computing is extremely high, with a total electricity bill of 9 million dollars per year. Thus, conserving energy and increasing the energy efficiency of supercomputers has become critical in recent years.

This dissertation proposes new solutions to address the above three key challenges for current large-scale storage and computing systems. Firstly, we propose a novel power management scheme called MAR (model-free, adaptive, rule-based) in multiprocessor systems to minimize the CPU power consumption subject to performance constraints. By introducing new I/O wait status, MAR is able to accurately describe the relationship between core frequencies, performance and power consumption. Moreover, we adopt a model-free control method to filter out the I/O wait status from the traditional CPU busy/idle model in order to achieve fast responsiveness to burst situations and take full advantage of power saving. Our extensive experiments on a physical testbed demonstrate that, for SPEC benchmarks and data-intensive (TPC-C) benchmarks, an MAR prototype system achieves 95.8-97.8% accuracy of the ideal power saving strategy calculated offline. Compared with baseline solutions, MAR is able to save 12.3-16.1% more power while maintain a comparable performance loss of about 0.78-1.08%. In addition, more simulation results indicate that our design achieved 3.35-14.2% more power saving efficiency and 4.2-10.7% less performance loss under various CMP configurations as compared with various baseline approaches such as LAST, Relax, PID and MPC.

Secondly, we create a new reliability model by incorporating the probability of replica loss to investigate the system reliability of multi-way declustering data layouts and analyze their potential parallel recovery possibilities. Our comprehensive simulation results on Matlab and SHARPE show that the shifted declustering data layout outperforms the random declustering layout in a multi-way replication scale-out architecture, in terms of data loss probability and system reliability by upto 63% and 85% respectively. Our study on both 5-year and 10-year system reliability equipped with various recovery bandwidth settings shows that, the shifted declustering layout surpasses the two baseline approaches in both cases by consuming up to 79 % and 87% less recovery bandwidth for copyset, as well as 4.8% and 10.2% less recovery bandwidth for random layout.

Thirdly, we develop a power-aware job scheduler by applying a rule based control method and taking into account real world power and speedup profiles to improve power efficiency while adhering to predetermined power constraints. The intensive simulation results shown that our proposed method is able to achieve the maximum utilization of computing resources as compared to baseline scheduling algorithms while keeping the energy cost under the threshold. Moreover, by introducing a Power Performance Factor (PPF) based on the real world power and speedup profiles, we are able to increase the power efficiency by up to 75%.

ACKNOWLEDGMENTS

This dissertation would not been possible without the help and support of a number of people. First and foremost, I would like to express my sincerest gratitude to my adviser, Dr.Jun Wang, for the tremendous time, energy and wisdom he has invested in my graduate education. His inspiring and constructive supervision has always been a constant source of encouragement for my study. I also want to thank my dissertation committee members, Dr.Ronald DeMara, Dr.Yier Jin, Dr.Shaojie Zhang and Dr.Liqiang Ni, for spending their time to view the manuscript and provide valuable comments.

I would like to thank my past and current colleagues: Pengju Shang, Huijun Zhu, Jiangling Yin, Xuhong Zhang, Dan Huang and Xunchao Chen. I want especially thank to Xuhong, for the inspiring discussion and continuous support to improve the quality of each work. A special thanks to Jiangling, for tremendous help on my experiment setups. My gratitude also goes to Pengju Shang, whos previous work provides great inspirations of this dissertation.

I dedicate this thesis to my family: my parents Yunfu Wang and Mingxia Zong, my husband Chun Tang, for all their love and encouragement throughout my life. Last but not least, I would also like to extend my thanks to my friends, who have cared and helped me, in one way or another. My graduate studies would not have been the same without them.

Last but not least, my works are supported in part by the US National Science Foundation Grant CCF-1527249, CCF-1337244 and National Science Foundation Early Career Award 0953946.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

We are now entering the Big Data era, in which a huge amount of data are generated and analyzed by data-intensive applications as well as the large-scale computing and storage systems. These are reflected in the following three aspects. In the current processor market, multicore processors or CMPs have become a mainstream because of the tremendous improvement in transistor density and the advancement in semi-conductor technology. At the same time, the limitations in instruction level parallelism (ILP) coupled with power dissipation restrictions encourage us to enter the "CMP" era for both high performance and power savings [1], [2]. On the other hand, an increasing number of big data storage systems are turning to use multi-way replication based storage architecture for the sake of high availability and reliability [62, 60, 48, 54]. As the storage system scale up, it increases both capacity and bandwidth, but it also makes disk failures more common [79]. Thirdly, as the demand for high-performance computing on supercomputers and clusters continues to grow, there will be a substantial increase in the total power consumption. In the LiveScience report, the supercomputer "Titan" costs 100 million dollars and its electricity bill is expected to cost about 9 million dollars per year [100]. Therefore, even if scientists can benefit from the fast computing speeds provided by supercomputers, we cannot neglect the staggering electricity bill associated with these computing service.

In a computing or storage system like this, three important factors need to be taken into account for design considerations which are power conservation,reliability as well as the energy efficiency. More specifically, we propose three new approaches to improve the above three metrics in large-scale computing and storage systems as shown in Figure 1.1.

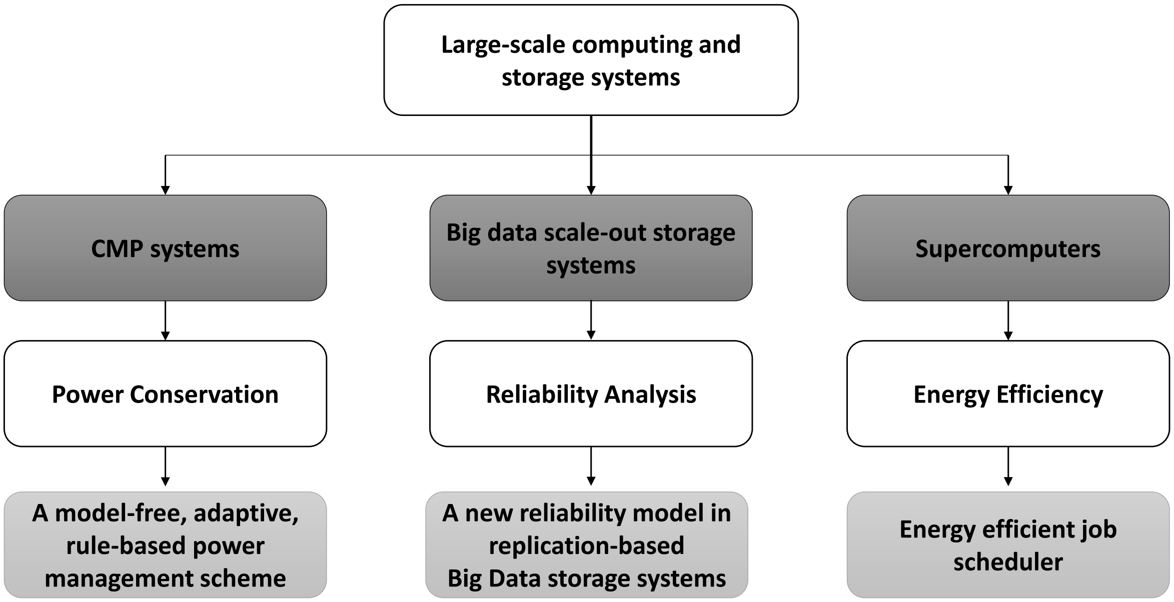

Large-scale computing and storage systems
CMP systems
Big data scale-out storage systems
Supercomputers
Power Conservation
Reliability Analysis
Energy Efficiency
A model-free, adaptive, rule-based power management scheme
A new reliability model in replication-based Big Data storage systems
Energy efficient job scheduler

Figure 1.1: Research work overview

MAR: A Novel power management scheme

In recent years, many power management strategies have been proposed for CMP processors based on DVFS [5], [6], [7]. DVFS (Dynamic Voltage and Frequency Scaling) is a highly effective technique used to reduce the power dissipation. Previous research [8, 9, 10, 11] has successfully balanced the CPU power consumption and its overall performance by using chip-wide DVFS. However, this coarse-granular management cannot be directly applied to CMP chips where per-core DVFS is available for more aggressive power-saving. Several recently proposed algorithms [12], [13] are based on open-loop search or optimization strategies, assuming that the power consumption of a CMP at different DVFS levels can be estimated accurately. This assumption may result in severe performance degradation or even power constraint violation when the workloads vary significantly from the one they used to perform estimations. There are also some closed-loop solutions based on feedback control theory [14, 15] which obtain power savings by using detected statistics of each core. The most widely adopted CPU statistics are CPU busy/idle times [16]. This busy/idle time approach referred as B-I model works well for non-I/O intensive applications. Unfortunately, data Intensive applications are mainly I/O bound, which becomes the main focus in this work. They often exhibit the unpredictable nature of I/O randomness and thus make the CPU utilization status hard to track. Consequently, the B-I model might not work well at this circumstance.

It may be noted that the I/O operations could actually affect the processor's both performance and power consumption, especially when we deal with data-intensive applications. More specifically, by taking I/O factors into consideration, we are able to make best use of CPU slack periods to save power more effectively and more efficiently. One example is when CPU is waiting for I/O operations to complete. Through our experiments, we found out several facts that can be utilized to conserve more power for a CMP processor. First of all, each core's waiting time for a completed I/O operation can be separated from its idle or working status. Secondly, appropriate scaling down the core's frequency during its I/O wait could provide more opportunities to save power without sacrificing overall system performance. Thirdly, the core frequency does not directly affect the I/O wait which means no relationship exists between these two factors. According to the above-mentioned observations, we develop our power management solutions for data-intensive applications using the following specific ways:

1. Considerations of each core's I/O wait status besides its working and idle status.

2. Accurately quantifying each status for accurate power-saving decisions.

3. A precise description of the relationship between frequency, performance and power consumption when the I/O wait factor is considered.

When we integrate the I/O wait into the power control system design, one challenge lies that the CPU workload is unable to be modeled because of the I/O randomness, which mainly results from the diversity of I/O patterns and the internal characteristic of storage systems. In our experiment, we found that even with a comprehensive control policy, such as MPC, it is impossible to accurately predict the CPU workloads for I/O intensive applications. To resolve this issue, we employ a fuzzy logic control to simplify the model construction work, which incorporates a precise logic and approximate reasoning into the system design and obtains much more accurate representations [40, 41, 42, 43, 44].

We develop a multi-input-multi-output power management system named MAR (model-free, adaptive, and rule-based) for CMP machines. It adopts a fuzzy logic for the power management by considering the I/O wait rather than employing traditional B-I model. Our design could precisely control the performance of a CMP chip at the desired set point while save 12.3-16.1% more power at run-time. Moreover, we are able to save even more power if the loaded CPUs support more frequency levels, as detailed in the experimental results in Chapter 3.

There are four primary contributions of this work:

- Developing a model-free, adaptive and rule-based power management method named MAR.
- Achieving fast-responsiveness through fast detecting methods that adopted by MAR.
- Conducting extensive experiments to examine the impacts of the cores seven working status and proposing an accurate description about the relationship between frequency, performance and power consumption.
- Simplifying the power management method by introducing fuzzy control theory in order to avoid the heavy relying on precise system model.

Reliability Analysis

Today, an increasing number of big data storage systems are turning to use multi-way replication based storage architecture for the sake of high availability and reliability [62, 60, 48, 54].

Reliability is increasingly important in big data storage systems built from thousands of individual components. As the storage system scale up, it increases both capacity and bandwidth, but it also makes disk failures more common [79]. In petabyte-scale file systems, disk failures will happen in a daily bases. In the system with thousands machines and storage devices, even in-frequent failures become likely. Therefore, analyzing the system reliability through an analytic method is useful since they can provide a reference for developers to choose the best layout catering their requirements.

With regards to the modeling of system reliability, it should considers all the situations that will make the system unreliable. Intuitively, when people thinking about system reliability, they will tend to consider the data loss probability and the system recovery bandwidth [50, 51, 81]. However, in a replication-based storage systems, the loss of replicas will also have an influence on the system reliability which should not be neglected. This motivates us to create an effective and comprehensive analytic model to obtain an accurate reliability results.

It may be noted that it is not feasible to build a real testbed to evaluate the reliability of a large-scale redundancy storage system with different layout schemes. Once the block distribution scheme is selected, the layout can no longer be changed after the production storage system is up to work. Moreover, there is no systemic method which can be used to quantify the reliability for multi-way replication based data placement methods. This is especially true when the number of copies exceeding two. This is because the replicas of data on one failed disk are distributed among multiple disks in the declustered storage system, if either one of those disks fails before the failed disk completely recovered, the data block will lost permanently. As a result, the data loss probability has increased. This motivates us to exploit the impact of data layouts on reliability in multi-way replication storage systems.

We propose a comprehensive reliability model to investigate the system reliability of multi-way declustering data layouts and analyzing their potential parallel recovery possibilities.

Specifically, we make the following contributions:

1. Propose a new reliability model to investigate the system reliability of multi-way declustering data layouts.

2. Utilize the reliability model for analyzing the data loss probability and system repair rate with different data layout schemes;

3. Quantify the most important parameter used in the model, $P^{(l)}$, which is the probability that the disk does not lose data with l failed disks in the system with either mathematical proof for copyset replication layout [46] or reasonable sampling techniques for shifted declustering [80] and random declustering[1];

4. Analyze the possible optimal parallel recovery schemes for copyset replication, shifted declustering and random deccultering layouts;

5. Make specific comparison between these three layouts, which are the most widely used in current enterprise large-scale multi-way replication storage systems;

6. Simulate the reliability model, compare and explain the system reliability with considering of various recovery bandwidth.

[1] Random declustering layout distributes data blocks according to given randomization algorithms, which map the key (or the index) of a data block to a position in the storage system.

Power-aware job scheduling

Supercomputers and clusters today are usually managed by batch job scheduling systems, which partition the available set of compute nodes according to resource requests submitted through job scripts and allocate the partitions to parallel jobs [86]. As the demand for high-performance computing on supercomputers and clusters continues to grow, there will be a substantial increase in the total power consumption. In the LiveScience report, the supercomputer ”Titan” costs 100 million dollars and its electricity bill is expected to cost about 9 million dollars per year [100]. Moreover, the computing and storage components of data centers consume around 61 billion kilowatt hours kWh of power per year, which costs about 4.5 billion U.S dollars. The power consumption of supercomputers and Data centers combined is about 1.5$\%$ of the total output of 15 typical power plants [101, 102]. Therefore, even if scientists can benefit from the fast computing speeds provided by supercomputers, we cannot neglect the staggering electricity bill associated with these computing service. In this case, it is necessary to apply a power-capping strategy for large-scale data storage or computing systems.

Job scheduling is a critical task for large-scale systems. Advanced schedulers can help to improve resource utilization and quality of service. Therefore, an advanced job scheduler should satisfy the following requirements. First,it must satisfy the user’s requirements, such as job start time, requested number of processors, requested job finish time, and other priority constraints. Secondly, the scheduler should allocate the computation resources reasonably and effectively without exceeding the power limits. Thirdly, it should apply different speedup schemes while satisfying the user’s requirements and saving as much power as possible. Finally, multiple power constraints should be introduced to adjust the power limit based on the characteristic of incoming jobs to explore further opportunities for energy conservation.

Many researchers are looking into this problem and trying to balance the energy conservation and performance of supercomputers with various methods. For energy efficiency, some researchers attempt to schedule the job by applying Deadline Constraints on Data Grids or HPC systems [84, 89]. Some of them focus on developing energy-efficient multi-job scheduling models and algorithms for cloud computing [94, 93]. Others are proposed a technique to reduce the energy consumption by incorporating a predictive thermal model to select possible jobs to reduce the energy [87]. These algorithms are developed to conserve energy either by integrating the DVFS technique to examine the behavior of the workload or by introducing a thermal predictive model to minimize the energy needs. However, reducing energy consumption is not enough to alleviate the energy consumption issues of supercomputers. This work proposes to tackle the problem from the angle of improving the energy efficiency.

To the best of our knowledge, we present the first attempt to provide power efficient job scheduling on supercomputers by taking the power budget and speedup files into consideration. However, our method can provide opportunities to improve power conservation, which should not been neglected. In this work, we propose a novel rule-based power-aware job scheduler to efficiently schedule the incoming jobs while maintaining the power budget. The main contributions of this work are:

1. We develop a novel rule-based power-aware job scheduler to achieve the best system utilization while considering the power budgets and various speedup settings.

2. We generate a new scheduling algorithm to support our proposed job scheduler.

3. We create a series of rules to assist the scheduler with determining the appropriate power consumption as well as speedups.

4. We propose a new Power Performance Factor (PPF) to evaluate the power efficiency under a fixed power budget.

CHAPTER 2: LITERATURE REVIEW

In this section, I would like to discuss the current research and their limitations with regards to the power management and the system reliability for large-scale computing and storage systems.

Power Management Background

In recent years, various power management strategies have been proposed for CMP systems. From the perspective of DVFS level, previous power management schemes could be divided into two categories which are chip-level and core-level power management. Chip-level power management uses chip-wide DVFS. In chip-level management [8, 9, 10, 11], the voltage and frequency of all cores are scaled to the same level during the program execution period by taking advantage of the application phase change. These techniques extensively benefit from application "phase" information that can pinpoint execution regions with different characteristics. They define several CPU frequency phases in which every phase is assigned to a fixed range of Mem/op. However, these task-oriented power management schemes do not take the advantage from per-core level DVFS. Core-level power management means managing the power consumption of each core. [12] and [13] collect performance-related information by an on-core hardware called performance monitoring counter (PMC). There are several limitations by using PMCs: Each CPU has a different set of available performance counters, usually with different names. Even different models in the same processor family can differ substantially in the specific performance counters available [38]; modern super scalar processors schedule and execute multiple instructions at one time. These "in-flight" instructions can retire at any time, depending on memory access, cache hits, pipeline stalls and many other factors. This can cause performance counter events to be attributed to the wrong instructions, making precise performance analysis difficult or impossible. Some recently proposed power management [16] approaches use MPC-based control model, which is derived from cluster level or large scale data centers level power management, such as SHIP [33], DEUCON [39], [15] and [16]. They make an assumption that the actual execution times of real-time tasks are equal to their estimated execution times, and their online-predictive model will cause significant error in spiky cases due to slow-settling from deviation. Moreover, their control architecture allows degraded performance since they do not include the performance metrics into the feedback. [35] tries to satisfy QoS-critical systems but their assumption is maintaining the same CPU utilization in order to guarantee the same performance. However, this is not true for the CPU unrelated works, such as the data-intensive or I/O-intensive workloads. Rule-based control theory [40] is widely used in machine control [41, 42], which incorporating the precise logic and approximation reasoning into the system design and obtaining much more accurate representations. More specifically, Rule-based control logic can be viewed as an attempt at formalization of the remarkable human capabilities. It is not "fuzzy" but a precise logic of imprecision and approximate reasoning with four distinguish features, which are graduation, granulation, precisiation and the concept of a generalized constraint [44]. It also reduces the development time/cycle, simplifies design complexity as well as implementation, and improves the overall control performance [43].

Reliability Analysis in Storage systems

Copyset replication

Copysets [46] replication technique is proposed to reduce the data loss frequency by introducing an near optimal trade off between number of nodes on which the data is scattered and the possibility of data loss. The problem is defined by several parameters R,N,S, which stand for number of replicas, number of nodes and the scatter width respectively. There are two phases of the copyset application technique which are permutation and replication. The first phase refers to an offline activity that creates a number of permutations by randomly permuting the nodes into the system. The second phase executes when a chunk needs to be replicated. The number of permutation is depends on scatter width S, which is equal to $S/(R-1)$. The primary replica can be placed on any node of the system, while others are placed on the nodes of a randomly chosen copyset that contains the first node. This design provides an optimal tradeoff between the scatter width and the number of copysets. Comparing with random replication method, copysets is able to reduce the probability of data loss form 99.99% to 0.15% under power outage in a 5000-node RAMCloud cluster.

Random and shifted declustering approach

Recently, A general multi-way replication-based declustering scheme has been widely used in enterprise large-scale storage systems to improve the I/O parallelism. Specifically, a random replication method [48] is widely used in large-scale storage systems due to its simple replication technique and strong protection against uncorrelated failures. Unfortunately, its high data loss rate and poorly handling of common correlated failures make the random replication method lose its generality in large-scale storage systems. There is also another copyset replication method [46] that has been proposed to obtain lower data loss frequency. Although the copyset replication technique is able to greatly reduce the probability of data loss, however it is trades off the data loss frequency with the amount of lost data in each incident, thus reliability issue remains a main concern. To improve the system reliability, a placement-ideal data layout–Shifted declustering [80] has been created to obtain optimal parallelism in wide variety of configurations and load balancing in both fault-free and degraded modes.

Muti-way Replication

RAID is first introduced in the mid 80's for improving both the performance and reliability of storage systems by providing redundancy through replication or parity schemes. RAID-1 applies mirroring, the simplest policy for replication based redundancy. Declustered layout schemes for replication based disk arrays include chained declustering [67], group-rotational declustering [58], and interleaved declustering [59]. Among them, chained declustering can be generalized to multi-way replication, but to the best of our knowledge, no such implementation exists. Group-rotational declustering is used in media streaming servers configured with 3-way replication to improve the quality of Video On Demand services [57]. Interleaved declustering is difficult to leverage up to multi-way replication. shifted declustering is a scalable solution for multi-way replication storage over any number of disks achieving optimal parallel performance [80].

Due to the availability benefits and the ease of maintenance, the switch from parity to multi-way replication is being further encouraged by the wide adoption by such mission critical systems as Google's File System (GFS) [62], Amazon's Dynamo [60] used to power Amazon.com and Amazon's S3 service, Microsoft's FARSITE [54], projects using Apache's Hadoop File System (HDFS) [48], video on demand services [57], and Geographic Information Systems (GIS) [74].

There are also theoretical methods for replica distribution in a large data cluster. RUSH [66] was introduced for the replica placement issues in large distributed storage systems. RUSH is a family of algorithms that have excellent randomization performance and low conflict probabilities, so it can be used in distributed storage systems which allowed each node to distribute data objects. Tosun compared a series replicated declustering schemes for spatial data systems such as GIS, where majority queries obey some explicit patterns [74], for example, range queries cover a geometric shape of a spatial data set. However, decentralized systems have not been used in reality. Current enterprise systems such as the Google File System [62], are still centrally controlled due to the ease of maintenance.

Existing Reliability Models

Xin *et al.* [78, 79] analyzed the reliability in terms of MTTDL (mean time-to-data-loss) of large-scale distributed storage systems configured with two-way and three-way replication. They apply Markov chains, where the state represents the failed number of data objects in a single redundancy group[1]. This model is based on the assumption that the data objects are independent, as well as redundancy groups. With this assumption, it makes sense to only model one redundancy group is enough, and if at time t, the reliability of one redundancy group is $R(t)$, then the reliability of the whole system is simply $1-(1-R(t))^r$, where r is the number of redundancy groups in the system. In this model, single data object failure should be allowed. However, they take disk failure rate as the data object failure rate. With this failure rate, all data objects on a disk fail when the disk fails, accordingly the failure of data objects are no longer independent, and it is contradictory to the assumption. As long as the events of disk failure exist, it is not reasonable to view the redundancy groups as independent. We will also explain that disk failure is the dominant factor rather than data block failure to evaluate the impacts of data layouts on system reliability in Section 4. Gafsi *et al.* applied continuous time Markov chains (CTMC) to model the reliability of distributed video servers [61]. Their analysis is based on parity redundancy and two-way replication redundancy. They categorized the redundancy schemes into one-to-one, one-to-all and one-to-some, where one-to-all and one-to-some are declustering layout policies.

Currently, there is no systematic research on the reliability issues for redundancy systems with multi-way replication. Among reliability metrics, the most straightforward one to understand is

[1] For replication based redundancy, a *redundancy group* includes all copies of a data unit. For parity based redundancy, a redundancy group includes all data stripe units as well as parity units of a data stripe.

the failure free period, during which the system provides services without any data loss.

CHAPTER 3: MAR: A NOVEL POWER MANAGEMENT SCHEME FOR CMP SYSTEMS IN DATA-INTENSIVE ENVIRONMENT

Learning Core's Behaviors

In this section, we exploit the behaviors of each core in a CMP processor to learn the relationship between power consumption, performance, and frequency settings, as shown in Figure 3.1. As widely shown in previous works, CPU power consumption and performance are both highly related to CPU frequency [5], [19]. Our experiment results demonstrate that there exist a cubic relationship between power consumption and CPU frequency which is well documented and shown in Figure 3.1. However, the relationship between performance and frequency is difficult to be modeled: the same frequency setting may result in a different response time (rt) or execution time (et) for various types of applications. Hence, the performance is related to both the processor's frequency and the workload characteristics. On the other hand, the behavior of the CPU is able to illustrate the characteristics of the running workloads. More specifically, each core in a CMP has seven working statuses, which we denote as the "metrics" in the rest of this work:

- user: normal processes executing in user mode;
- nice: niced processes executing in user mode;
- system: processes executing in kernel mode;
- idle: idle times;
- I/O wait: waiting for the I/O operations to complete;
- irq: servicing interrupts;

- softirq: soft servicing interrupts;

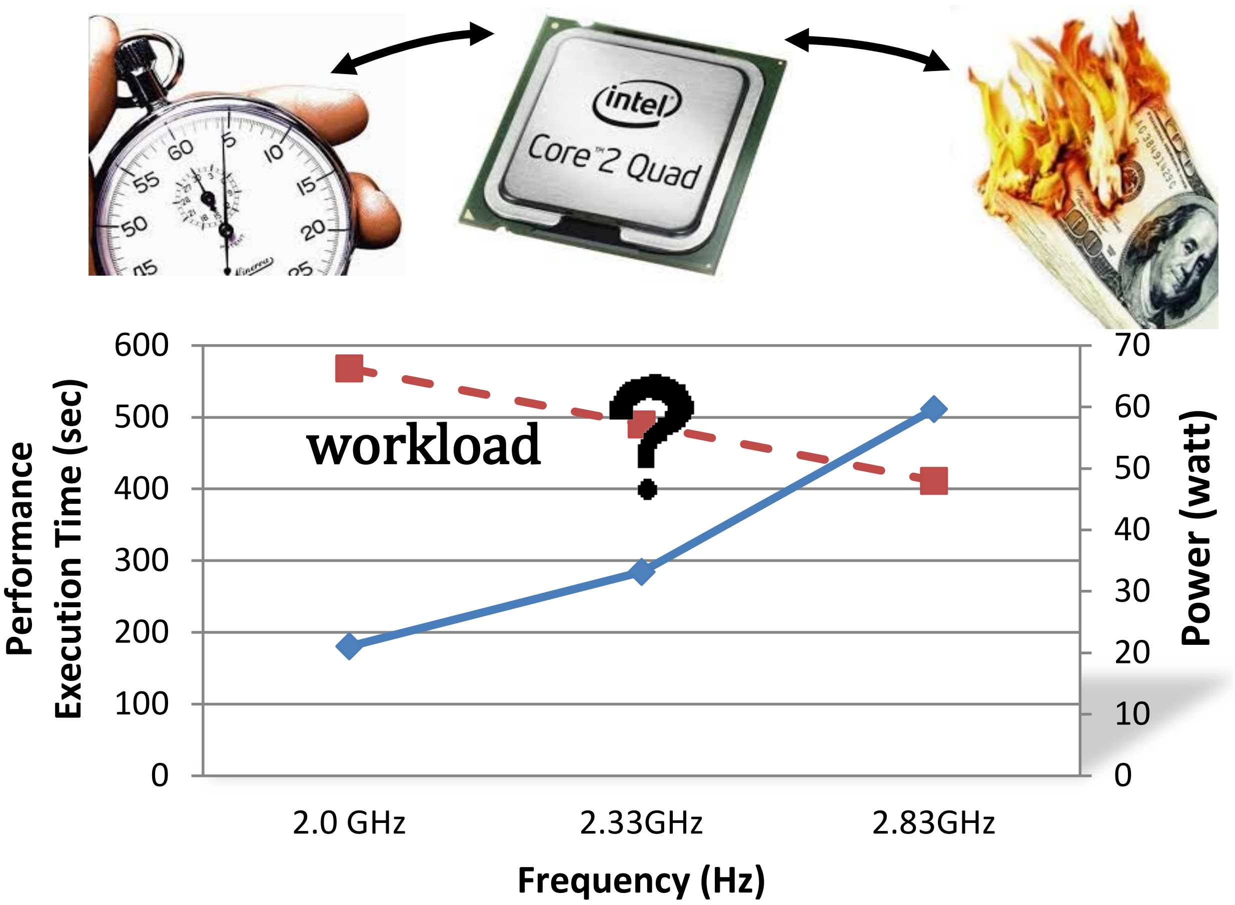


Figure 3.1: The relationship between CPU's frequency, power consumption and performance

The duration of the cores seven statuses completely exhibit the composition of the running workload. As a result, the performance is determined by a function F with considering both of the CPU frequency and the workload characteristic. Since the workload can be formulated using ϕ which is a function of the seven working statuses, the system performance can be present in Equation (4.9).

$$\begin{aligned} et = {} & F(frequency, workload) \\ = {} & F\{f, \phi(metrics)\} \\ = {} & F\{f, \phi(user, nice, sys, idle, iowait, irq, softirq)\} \end{aligned} \tag{3.1}$$

We launch various applications on our testbed to learn the curve of Equation (4.9), e.g. an I/O bomb from Isolation Benchmark Suite (IBS) [22], a gcc and a mcf benchmark from SPEC CPU2006 suite version 1.0 [23], and TPC-C running on PostgreSQL [24]. I/O bomb uses the IOzone benchmark tool to continuously read and write to the hard disk (by writing files larger than main memory to ensure that it is not just testing memory); mcf is the most memory bound benchmark in SPEC CPU2006 while gcc is a CPU-intensive application, as shown in Table 3.1. TPC-C is a standard On-Line-Transaction-Processing (data-intensive) benchmark. The configuration details of these benchmarks could be found in Section 3. We run these tests on a Quad-Core Intel Core2 Q9550 2.83 GHz processor with 12MB L2 cache and 1333 MHz FSB. The CPU supported frequencies are 2.0 GHz, 2.33 GHz, and 2.83 GHz.

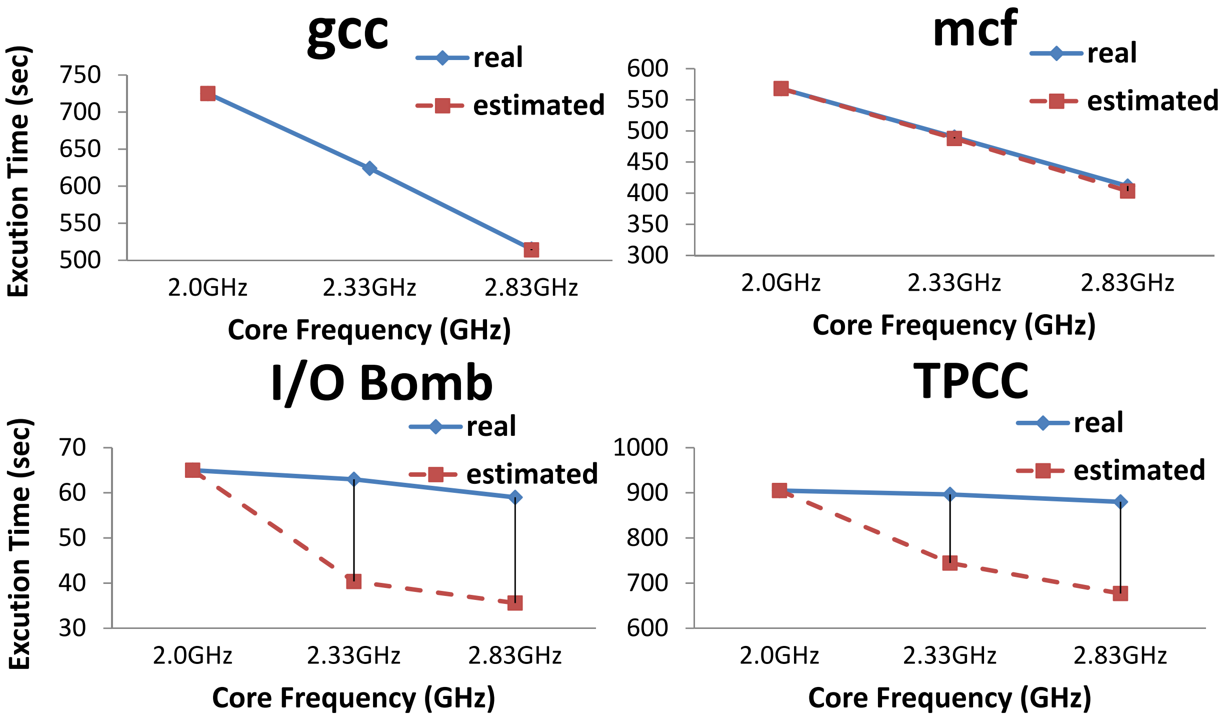


Figure 3.2: Overall execution time of different workloads

Per-Core

Because we are using per-core level DVFS for power management, it is necessary to understand the meanings of the 7 statuses of each single core. In previous works, the relationship between CPU behavior and estimated performance et is simply determined by the CPU busy/idle (B-I) time [14], [25]. This B-I model is defined as Equation (3.2):

$$et_{new} = et_{old} * (\frac{P_{busy} * f_{old}}{f_{new}} + P_{idle})$$
$$P_{busy} = \frac{(\frac{rt_{new}}{rt_{old}} - P_{idle}) * f_{new}}{f_{old}} \tag{3.2}$$

where P_{busy}[1] is the busy ratio of CPU while P_{idle} is the idle percentage of CPU; f_{old} and f_{new} are the two version of CPU frequency settings. We first enable only one core in the CMP processor and assign one process to run the benchmarks, so that we can avoid the noise from task switches among the cores. The upper two charts in Figure 3.2 illustrates that for the first two workloads, e.g. gcc (CPU-intensive) and mcf (memory-intensive), the B-I model is accurate enough with less than 3% deviation. However for the I/O intensive or data-intensive workloads, e.g. I/O bomb and TPC-C showing in the lower two charts in Figure 3.2, using the B-I model which does not consider the I/O impact will result in up to 39.6% modeling errors. The reason that B-I model works well for CPU-intensive and memory-intensive workloads is because of Instruction-Level Parallelism (ILP). The latency caused by cache misses, and mis-predictions will be eliminated by advancing the future operations. However, ILP is not always capable of eliminating the latency caused by I/O operations [3], which leads to the prediction errors for I/O bomb and TPC-C benchmarks.

We also show the statistics of the 3 main working statuses in Figure 3.3. For gcc and mcf, most of the execution time is in user mode; the cache misses and mis-predictions of mcf have negligible impact on the CMP's behavior due to ILP. For I/O bomb, I/O wait is the main latency; for data-intensive benchmark TPC-C, the lower frequency will hide some of the I/O wait latency because of ILP, but the latency in both user and iowait modes cannot be ignored. For all four cases, the irq and softirq latency are very small which only constitute about 0.2% of the total working status. Therefore, irq and softirq will not be taken into account in our experiments since the latency cause by them cannot affect the overall system performance comparing with other major latency. As a result, "user+nice+sys", "idle" and "I/O wait" are the three most important working statuses which could describe the CMP's behavior in general. Without considering I/O wait latency, the basic B-I model may result in non-trivial modeling errors for data-intensive or I/O intensive applications.

[1] The "Busy Time" in previous works is usually calculated as the equation of $overall/timeidle/time$ without the consideration of the cores other metrics [14], [25].

Table 3.1: Cache Miss Rate

Benchmarks	L1 Data Miss	L2 Miss	Mispredictions
gcc	14.21	3.17	5.11
mcf	130.15	36.73	15.79

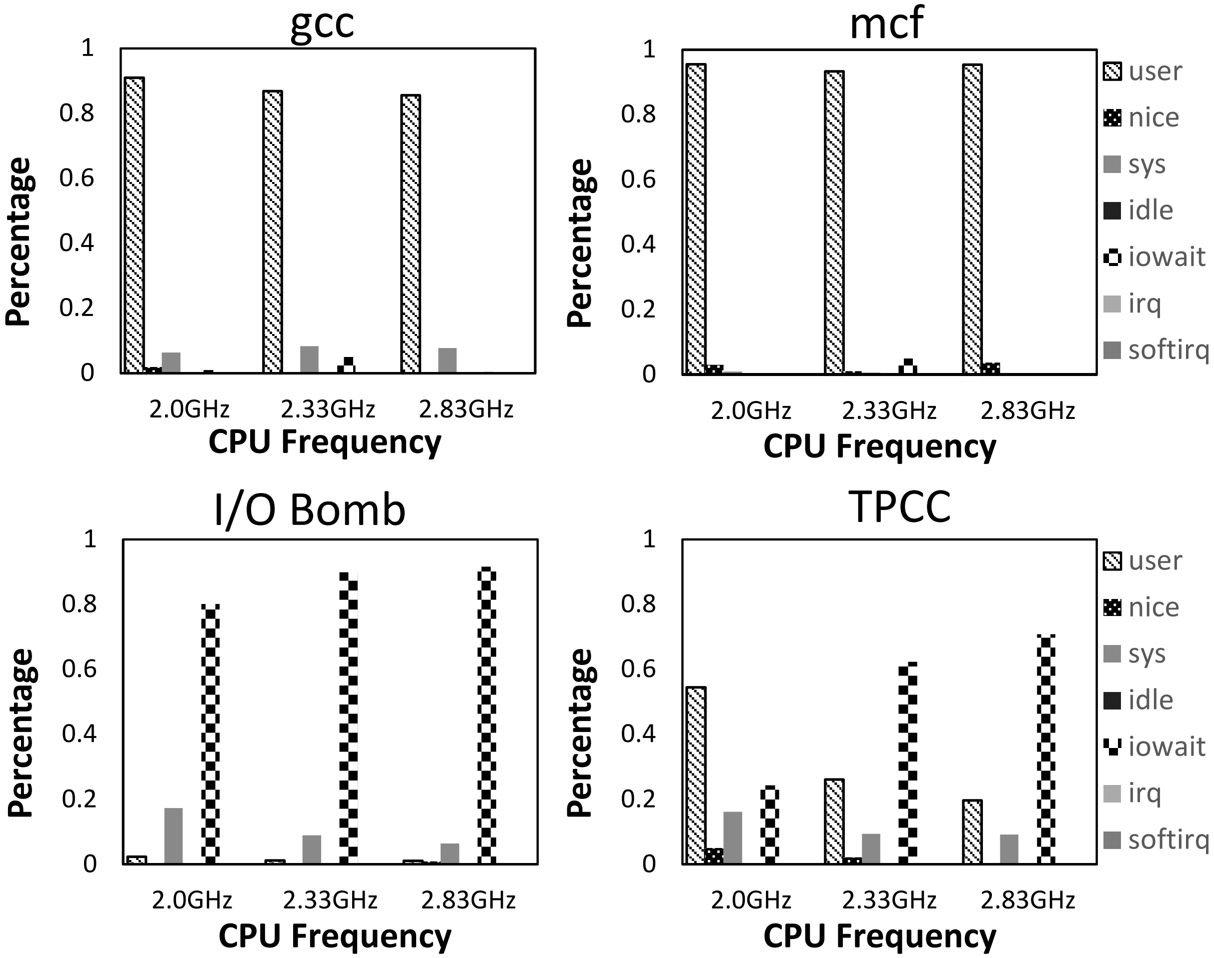


Figure 3.3: Core’s major statistics under different workloads

Because of the job scheduler in a CMP processor, one task in CMP processor may be switched among the cores during its run. In order to show whether this core-level task switches can eliminate the I/O wait latency, we run 7 processes on all 4 cores in our testbed. Each process will randomly run one of the following benchmarks: gcc, mcf, bzip2, gap, applu, gzip and TPC-C. Each core has 3 available frequency settings: 2.83 GHz, 2.33 GHz and 2.0 GHz.

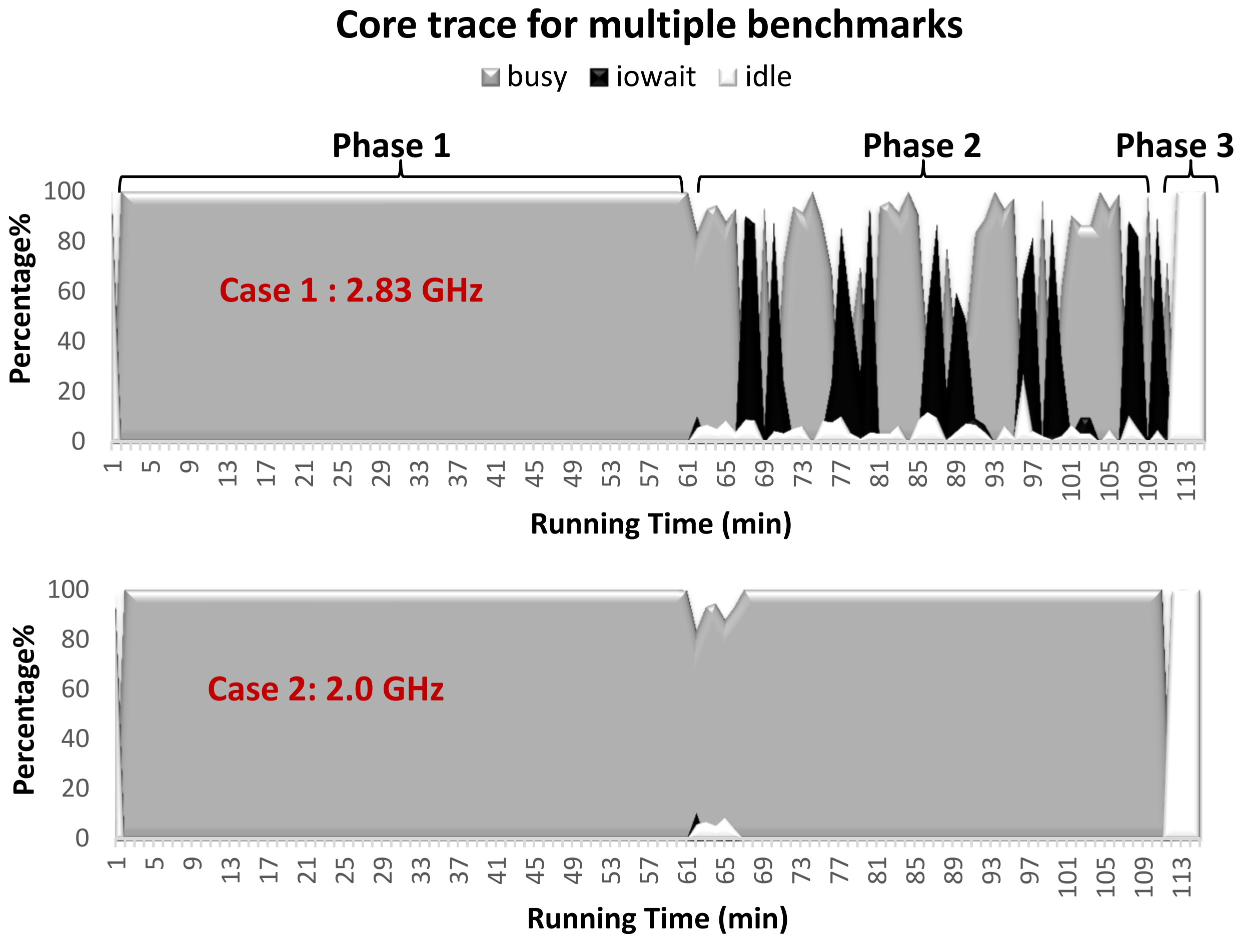


Figure 3.4: The trace of core0 under both high and low frequency; the overall execution times are comparable for the two cases.

Figure 3.4 shows the traces for core0 under different frequency settings. We omit “irq” and “softirq” based on the results of section 3, and we treat “user, nice, sys” as a group denoting the real “busy” status. When the frequency is 2.83 GHz, all the workloads are processed in parallel in “phase 1”; the I/O wait latency could be hidden by the process-level parallelism. However in “phase 2”, when there are not enough available processes to schedule, the I/O wait latency will emerge. After all processes are finished, the core will stay idle in “phase 3”. The traditional B-I based power management scheme is only able to discover the chances to save power in “phase 3” by lowering the processor’s voltage and frequency. However in fact, “phase 2” also provides opportunities to save more power. In this case, we can lower the frequency in order to parallel the CPU and I/O operations as much as possible.

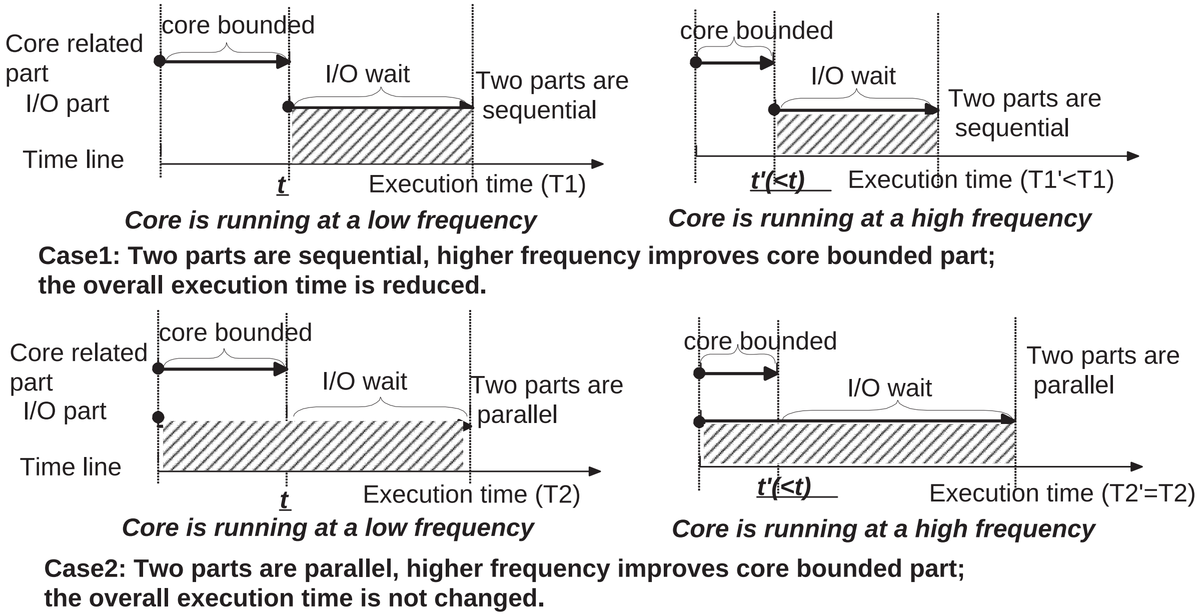

Core related part
I/O part
Time line
core bounded
I/O wait
Two parts are sequential
t
Execution time (T1)
Core is running at a low frequency
core bounded
I/O wait
Two parts are sequential
t'(<t)
Execution time (T1'<T1)
Core is running at a high frequency
Case1: Two parts are sequential, higher frequency improves core bounded part; the overall execution time is reduced.
Core related part
I/O part
Time line
core bounded
I/O wait
Two parts are parallel
t
Execution time (T2)
Core is running at a low frequency
core bounded
I/O wait
Two parts are parallel
t'(<t)
Execution time (T2'=T2)
Core is running at a high frequency
Case2: Two parts are parallel, higher frequency improves core bounded part; the overall execution time is not changed.

Figure 3.5: Two cases when I/O wait exits

In Case 2, as shown in the lower part of Figure 3.4, we can use “2.0 GHz” to process all the workloads roughly at a comparable execution time while only consumes 35.3% power as compared to the Case 1 that runs at the 2.83 GHz frequency. We note that heavy disk utilization may not necessarily result in I/O wait if there are enough parallel CPU-consuming tasks. Interestingly, the new data-intensive analyses and applications will incur long I/O wait latency [18, 26, 27, 28, 29]. As a result, I/O wait still needs to be taken into account in the big data era.

Analysis and Preparation for MAR Design

Although the I/O wait is the duration that the processor is waiting for the I/O operation to be finished, we cannot simply consider it as a sub-category of the idle time. Since if and only if CPU idle time exists, increasing the CPU frequency will linearly decrease the execution time. However, when I/O wait is present, there are two cases as shown in Figure 3.5. In Case 1 where the CPU-consuming tasks and I/O tasks are running asynchronously or blocking each other [30], the traditional B-I method can be utilized (discussed in Section 3) to model the relation between execution time and CPU frequency. In Case 2 where the two types of workloads are running in parallel but not well aligned, using the B-I model to scale CPU frequency will not affect the overall execution time. In other words, the traditional "B-I" model will not work precisely under this situation. In order to distinguish these two cases, we introduce two thresholds which are "th_{up}" and "th_{down}". "th_{up}" stands for the CPU scaling up threshold, and the "th_{down}" stands for the CPU scaling down threshold. With their help, we can quantify the Equation (4.9) as the following Equation (3.3). In the Case 1, the core is either in busy-dominate ($\omega < th_{up}$)or in idle-dominate ($\omega > th_{down}$), thus the traditional B-I model can be utilized. In the Case 2, the core is neither in busy nor in idle status, thus scaling the CPU frequencies will not affect the overall execution time. Therefore, the ratio of $\frac{rt_{new}}{rt_{old}}$ will be set to "1".

$$
\begin{aligned}
Case1: \quad & if\omega < th_{up}\text{when scaling up OR} \\
& if\omega > th_{down}\text{when scaling down:} \\
& P_{busy} = \frac{(\frac{rt_{new}}{rt_{old}} - P_{idle}) * f_{new}}{f_{old}} \\
Case2: \quad & Otherwise: \\
& P_{busy} = \frac{(1 - P_{idle}) * f_{new}}{f_{old}} \qquad (3.3)
\end{aligned}
$$

where P_{busy} represents the busy ratio; rt stands for response time; P_{idle} means the idle ratio. The default value of "th_{up}" and "th_{down}" are based on our comprehensive experimental results. Note that these two thresholds are affected by the throughput of I/O devices, L1/L2 cache hit rates, network traffic, etc.. A self-tuning strategy for these two thresholds will explain in detail in Section 3. Equation (3.3) can be used to complete the relationship among performance, frequency and the various types of workloads that we try to present in Figure 3.1. Our rule-based power management controller MAR will be designed according to the two relationships in Equation 3.3.

MAR's Design

In this section, we introduce the design, analysis, and optimization of our rule-based power management approach. In previous sections, we explain why I/O wait should be considered into power-management. One challenge is, when considering the I/O wait, the CPU workload will become unpredictable due to the I/O randomness. Therefore, the short sampling period that used to work well for the DVFS control might cause severe problems to the system, such as instability and noise. To attack this problem, we need to prolong the sampling period but not significantly stretch the response time under the variation of CPU utilization. There is a need for incorporating a thorough understanding of the control object and control system into MAR system design. Interestingly, a fuzzy logic control method is an ideal candidate solution here as it utilizes human's experience as input and enforce a precise logic in the control system to reflect the thorough understanding.

MAR Control model

The fuzzy logic has been used as the main base for the MAR's design, which includes fuzzification, rules evaluation and defuzzification. MAR is designed as a MIMO controller shown in Figure 3.6. In order to demonstrate that our control efficiency is better than any other control methods, we divide our control algorithm into two parts. In the first part, the I/O wait is not taken into account in order to prove that, the outcome of our control algorithm is more precise and the response time is faster. Secondly, as we define that the I/O wait is critical for power-saving strategy especially when running data-intensive tasks. Thus, I/O wait would be considered in the second part to show that our new "B-W-I" model can work accurately and efficiently. By introducing the I/O wait, Equation 3.3 can be further updated in the following equation (3.4):

$$\begin{aligned}
Case1: \quad & if\omega < th_{up}\text{when scaling up OR} \\
& if\omega > th_{down}\text{when scaling down:} \\
& cb = \frac{(\frac{rt}{RRT} - P_{idle}) * f_{new}}{f_{old}} \\
Case2: \quad & Otherwise: \\
& cb = \frac{\omega * f_{new}}{f_{old}}
\end{aligned} \tag{3.4}$$

where ω stands for the I/O wait ratio. Let SP denote the sampling period, RRT means the required response time, which is a key factor used to determine whether the system performance has been achieved, cb is core boundness of the workloads (the percentage of core's busy time compared with the sample time), ω is I/O wait ratio and ecb is the tracking error of core boundness. One basic control loop is described as follows: at the end of each SP, rt, cb, and ω vectors will feed back into the controller through an input interface. It should be noted that rt, ecb, and ω could be directly measured from last SP. These inputs will be processed into the arguments P_{busy}, ω, rt_{new} and rt_{old} of the Equation (3.3) or Equation (3.4) based on whether the I/O wait has been taken into account or not. Now we show how to calculate the arguments.

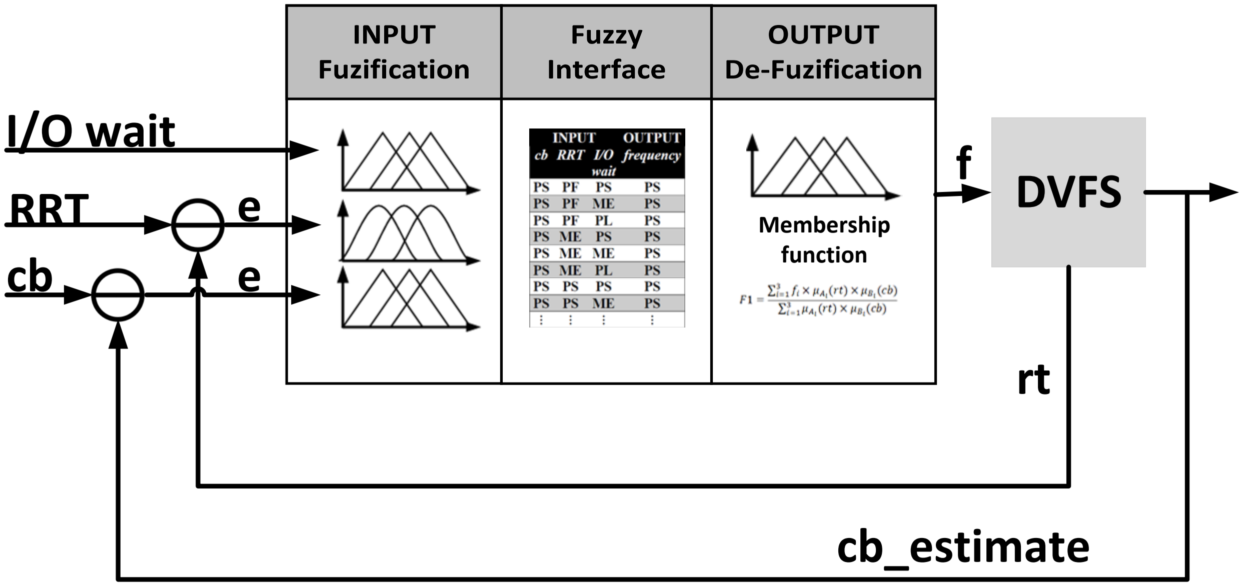
Fuzzification without consideration of I/O wait

We first fuzzify the input values of cb and rt by performing a scale mapping using membership function to transfer the range of crisp values into the corresponding universe of discourse. The universe of discourse means linguistic terms especially in Fuzzy Logic, such as “PS”, “ME” and “PL” represent for “positive short”, “moderate” and “positive large” respectively. Membership function represents the degree of truth as an extension of valuation. The response time rt is one input of the system that used to determine the performance of the controller. In order to set a performance constraint, we denote $0 < \delta < 1$ as the user-specified performance-loss constraint. We are using the symmetric triangular membership function that presented in Equation (3) for mapping between crisp value and linguistic terms. The reason that we apply a linear transformation to the response time is we have a specified minimum and maximum value in this case. Equation (3) present the membership function and Figure 3.7 plot of the function of rt.

$$\mu_A(rt) = \begin{cases} \dot{1} - \dfrac{rt - RRT}{RRT * \delta} & \text{,if} \mid rt - RRT \mid \leq RRT * \delta \\ 1 & \text{,if} rt > RRT(1 + \delta) \\ 0 & \text{,Otherwise} \end{cases}$$

(3.5)

where A stands for the fuzzy set of response time, which including {PF, ME, PS} stands for positive fast, medium and positive small. For each crisp value of response time, we can compute a set of μ that can be used in the defuzzification step by applying certain rules.



INPUT			OUTPUT
cb	*RRT*	*I/O wait*	*frequency*
PS	PF	PS	PS
PS	PF	ME	PS
PS	PF	PL	PS
PS	ME	PS	PS
PS	ME	ME	PS
PS	ME	PL	PS
PS	PS	PS	PS
PS	PS	ME	PS
⋮	⋮	⋮	⋮

$$F1 = \frac{\sum_{i=1}^{3} f_i \times \mu_{A_i}(rt) \times \mu_{B_i}(cb)}{\sum_{i=1}^{3} \mu_{A_i}(rt) \times \mu_{B_i}(cb)}$$

Figure 3.6: The architecture of MAR power management

Secondly, we fuzzify another input variable core boundness using Gaussian membership function because it transforms the original values into a normal distribution, which creates a smooth transformation rather than the linear functions. On the other hand, since the value of core boundness changes frequently, using Gaussian membership function can achieve the higher chances to detect the fluctuation and response accordingly. Equation (3.6) is the Gaussian membership function that used to map the crisp values for core boundness and Figure 8 shows the plot of the mapping.

$$\mu_B(cb) = \exp\left[-\frac{1}{2}\left(\frac{cb - cb_0}{\sigma}\right)^2\right] \qquad (3.6)$$

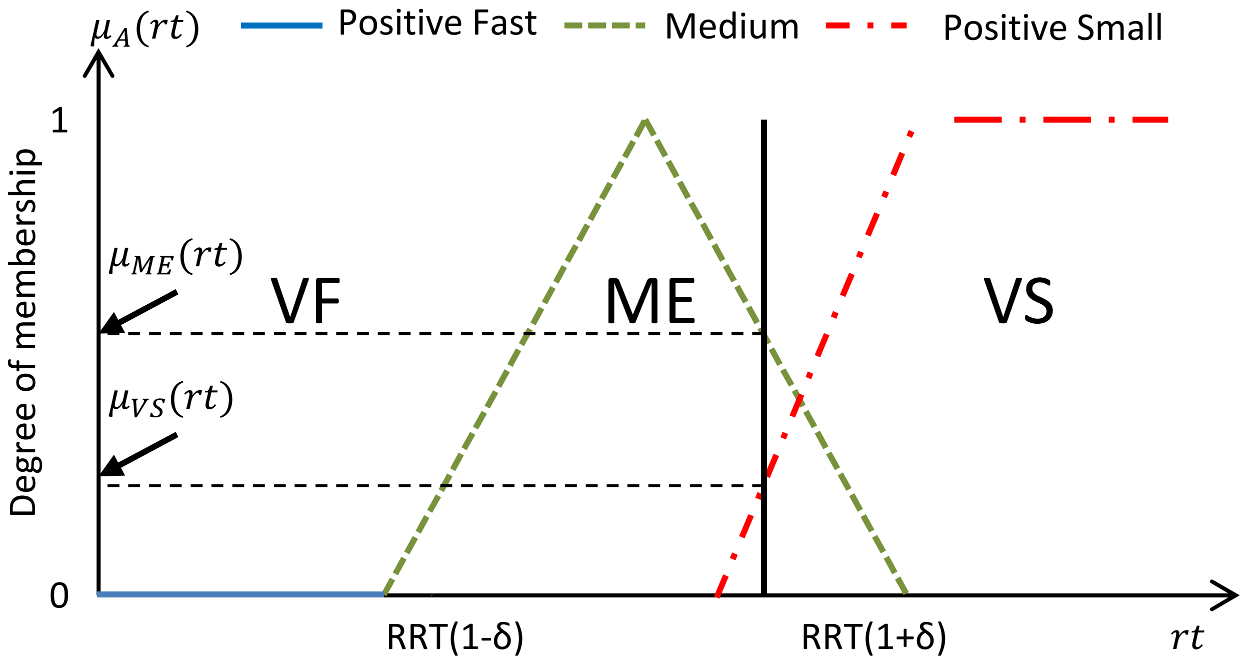


Figure 3.7: Mapping results for response time using Triangular Membership Function

Where B represents the fuzzy set of core boundness that including {PS, ME, PL} represent "Positive Small", "Medium" and "Positive Large"; cb_0 is the position of the peak relative to the universe and δ is the standard deviation. Figure 3.8 show that if we have a crisp value of core boundness falls between 0.4 and 0.5, we can calculate the degrees under each element of the fuzzy set.

Fuzzification with consideration of I/O wait

In the section, we will focus on the fuzzification of I/O wait. As we mentioned above, there are two thresholds in Section 3 that have been introduced to distinguish the parallelism of I/O wait and core boundness. Therefore, these two thresholds will be used as the minimum and maximum value for the mapping by utilizing symmetric triangular membership function. Equation (7) shows the relationship between crisp values and the membership degrees and Figure 3.9 present the plot accordingly.

$$\mu_C(\omega) = \begin{cases} 1 - \dfrac{\omega - (th_{down} + th_{up})/2}{(th_{up} - th_{down})/2} & ,\text{if} th_{down} < \omega < th_{up} \\ 1 & ,\text{if}\omega > th_{up} \\ 0 & ,\text{Otherwise} \end{cases}$$

(3.7)

where C represents the fuzzy set of I/O wait that including {PS, ME, PL} represent "Positive Small", "Medium" and "Positive Large"; th_{up} and th_{down} denote for the two thresholds.

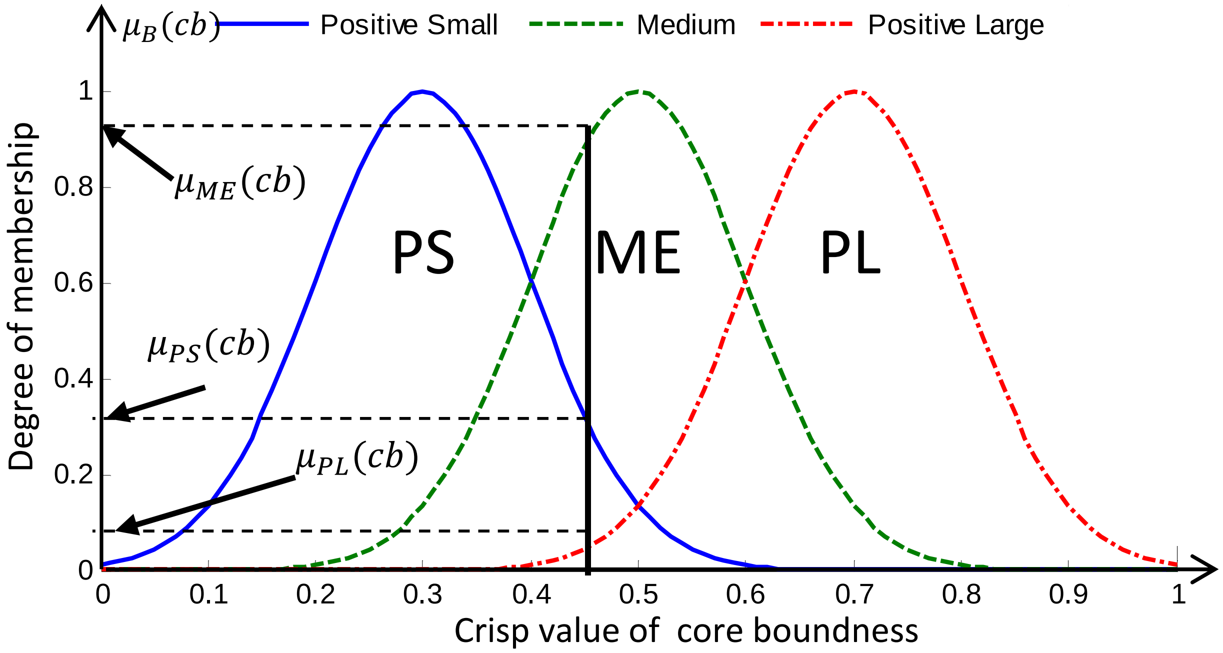


Figure 3.8: Mapping results for core boundness using Gaussian Membership Function

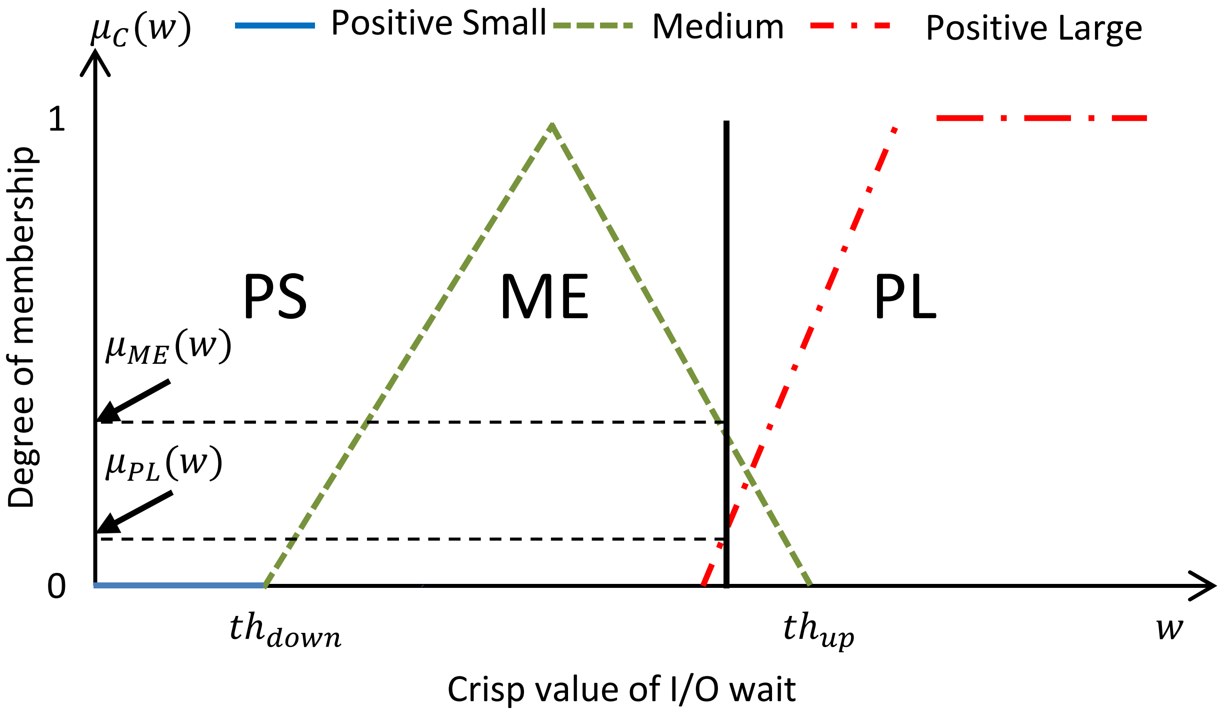


Figure 3.9: Mapping results for I/O wait using Triangular Membership Function

Fuzzy Rules

In this section, we propose a set of rules for MAR which will guide MAR to find the frequencies to be set in next Sampling Period. The fuzzy rules can be presented in the following two tables, which are Table 3.2 and Table 5.2. The Table 3.2 is about the fuzzy rules without considering I/O wait, which aims at demonstrating that our MAR control method works better than other existing control approaches.

Table 3.2: Fuzzy Rule Base(Part I)

INPUT		**OUTPUT**
core boundness	Response time	frequency
PS	PF	PS
PS	ME	PS
PS	PS	ME
ME	PF	PS
ME	ME	ME
ME	PS	ME
PL	PF	PS
PL	ME	ME
PL	PS	PH

Table 5.2 presents a set of fuzzy rules by taking I/O waits into account. The following paragraph provides a detailed explanation showing the procedure of generating the rules.

First, if $RRT(1+\sigma) \leq rt \leq RRT(1+\sigma)$: This is the ideal case from a performance perspective. Traditional solutions may not change the core's frequency setting. However MAR will do a further check whether $\omega > th_{down}$.

- If so, the frequency could be scaled down to a lower level to save more power without affecting the response time rt.
- If not, scaling the frequency will result in different response time rt, which is deviated from RRT. In this case, MAR will keep using the current frequency.

Secondly, if $rt > RRT(1+\sigma)$: This means the real response time does not meet the requirement, MAR checks whether $\omega > th_{up}$.

- If ω exceeds the scaling up threshold, changing to higher frequency will not improve the performance. Moreover, higher frequency will result in a higher I/O wait, which is a waste of core resources. As a result, MAR will keep the current frequency setting.

- If ω is within the threshold, a higher core frequency could improve response time rt in this case.

Finally, $rt < RRT(1-\sigma)$: If the measured response time is unnecessarily better than the requirement, there is a chance to scale the frequency down to save more power.

- If $\omega > th_{down}$, MAR will only scale down the core frequency by one level. The reason for this "lazy" scaling is because it is difficult to know what ω will be when using one level lower frequency. The new ω decides whether we should further scale down the frequency or not.
- If $\omega \leq th_{down}$ we may be able to scale down the core frequency to just meet the performance requirement while saving more power.

Centroid Defuzzification method

As we introducing two different kinds of membership functions which are triangular and Gaussian. There are also exist two types of centroid defuzzification methods. In the previous paragraph we present two fuzzification approaches for our MAR controller, with the difference in whether considering I/O wait or not. Equation (4.4) and Equation (3.9) were generated to deal with these two cases by using two kinds of defuzzification solutions.

$$F1 = \frac{\sum_{i=1}^{3} f_i * \mu_{Ai}(rt) * \mu_{Bi}(cb)}{\sum_{i=1}^{3} \mu_{Ai}(rt) * \mu_{Bi}(cb)} \tag{3.8}$$

$$F1 = \frac{\sum_{i=1}^{3} f_i * \mu_{Bi}(\sum_{i=1}^{3} min[\mu_{Ai}(rt), \mu_{Ci}(\omega)])}{\sum_{i=1}^{3} \mu_{Bi}(\sum_{i=1}^{3} min[\mu_{Ai}(rt), \mu_{Ci}(\omega)])}$$

(3.9)

where f_i stands for the center of CPU frequencies. In our case, f_i =2.0,2.33,2.83. Through Equation 4.4 and Equation 3.9, we are able to defuzzify the linguistic values that obtained in the output of the rule table by incorporating the various membership degrees. The output results will fall into three intervals which are $0 - 2.0$ GHz, $2.0 - 2.33$ GHz and $2.33 - 2.83$ GHz. Since the CPU only supporting 3 DVFS levels, the value below the first interval will automatically set to 2.0 GHz, otherwise, MAR will decide which frequency level will be set based on the output of the rule table. For example, if linguistic term of the output frequency says “PB”, 2.83 GHz would be set for the next sampling period.

Self-tuning Strategy

There are several factors affecting the thresholds th_{up} and th_{down} , for example:

1. Throughput of I/O devices. Higher I/O throughput means the same amount of data could be transferred in less “I/O wait” jiffies. As a result, the thresholds will become higher because the core needs more I/O wait jiffies to reach the time boundary, which defines whether core bounded part or I/O part is the determinant in execution time.

2. on-chip L1/L2 cache hit rate. Lower cache hit rate results in higher memory access, which is much slower than cache access. Therefore, the overall processing speed of the core bounded part (including both cache and memory access) becomes slower.

Table 3.3: Fuzzy Rule Base(Part II)

INPUT			**OUTPUT**
Core boundness	Response time	I/O wait	frequency
PS	PF	PS	PS
PS	PF	ME	PS
PS	PF	PL	PS
PS	ME	PS	PS
PS	ME	ME	PS
PS	ME	PL	PS
PS	PS	PS	PS
PS	PS	ME	PS
PS	PS	PL	PS
ME	PF	PS	PS
ME	PF	ME	PS
ME	PF	PL	PS
ME	ME	PS	PS
ME	ME	ME	PS
ME	ME	PL	PS
ME	PS	PS	ME
ME	PS	ME	ME
ME	PS	PL	PS
PL	PF	PS	PS
PL	PF	ME	PS
PL	PF	PL	PS
PL	ME	PS	ME
PL	ME	ME	ME
PL	ME	PL	PS
PL	PS	PS	PH
PL	PS	ME	ME
PL	PS	PL	PS

3. The noise in I/O wait, such as network I/O traffic file system journaling, paging swapping, etc.

4. Heat and heat dissipation. When processors run too hot, they can experience errors, lock, freeze, or even burn up. It is difficult to predict the thresholds in this case; hence we adopt

self-tuning methods based on the observed system behaviors. The self-tuning strategies are listed below:

- When $rt > RRT(1+\sigma)$, $\omega \leq th_{up}$, in this case, the RRT is not met and the frequency need to be scaling up for improving the performance. However, if the rt_{new} after the frequency scaling is same as the rt in last sampling period, we need to adjust it lower by Equation (3.10) :

$$th_{up_new} = th_{up_old} - \frac{\omega}{2} \tag{3.10}$$

- When $rt < RRT$, in this case, the RRT is met and the frequency may be scaled based on the rules presented in Section 3 . Thus, if $\omega \leq th_{down}$, RRT is over met and the frequency need to be scaled down, the th_{down} also need to be adjusted to a lower level. Else if $\omega > th_{down}$, RRT is over met and rt_{new} is changed which means ω should be lower than th_{down}, hence we set th_{down} to a higher level. Equation (3.11) can be used to set th_{down} either to a higher or lower level.

$$th_{down_new} = th_{down_old} + \frac{\omega}{2} \tag{3.11}$$

Methodology

In this section, we show our experimental methodology and benchmarks, as well as the implementation details of each component in our MAR controller.

Processor

We use a Quad-Core Intel Core2 Q9550 2.83GHz processor, with 12 MB L2 cache and 1333MHz FSB. The four execution cores are in four sockets. We change the DVFS levels of the 4 cores in each group together in order to have a real impact on the processor power consumption. Each core in the processor supports 3 DVFS levels: 2.0 GHz, 2.33 GHz and 2.83 GHz. The operating system is Ubuntu 12.04.3(LTS) with Linux kernel 3.8.13. For the extended experiment in Section 3, we use an Intel(R) Core(TM) 6700 2.66 GHz processor with two execution cores. Each core in the processor supports 5 DVFS levels, which are 1.33GHz,1.6GHz, 2.0GHz, 2.33GHz and 2.66GHz respectively. The operation system is Ubuntu 12.04.3(LTS) with Linux kernel 3.8.13.

Benchmark

We use 3 stress tests (CPU-bomb, I/O-bomb, and memory-bomb) from Isolation Benchmark Suite [22]; SPEC CPU 2006 suite version 1.0 [23], and data-intensive benchmark: TPC-C running on PostgreSQL [24]. TPC-C incorporates five types of transactions with different complexity for online and deferred execution on a data-base system. Every single transaction consists of computing part and I/O part. Due to the database buffer pool, the updating records will not be flushed until the pool is full.

Core Statistics

Various information about kernel activities are available in the /proc/stat file. The first three lines in this file are the CPU's statistics, such as usr,nice,sys,idle,etc.. Since we using the 3.8.13 version of Linus, the file includes three additional columns: iowait, irq, softiqr. These numbers identify the amount of time that the CPU has spent on performing different kinds of work. Time units are in

USER_HZ or Jiffies. In our x86 system, the default value of a jiffy is $10ms$, or 1/100 of a second. MAR needs to collect core boundness information as well as I/O wait latency. Each core's boundness is the sum of the jiffies in user, nice and sys mode divided by the total number of jiffies in last SP. Similarly, I/O wait latency is calculated based on the iowait column. The way to measure the real-time response time depends on the benchmarks. In Isolation Benchmark, the response time could be monitored by the I/O throughput. In TPC-C, the primary metrics, transaction rate (tpmC), could be used as the response time. However for SPEC CPU2006 benchmarks, it is difficult to find any metrics to denote response time because there is no "throughput" concept. Our previous experiments in Figure 2 show that these CPU-intensive and memory-intensive benchmarks have roughly linear relationships with core frequency. Hence we can calculate the number of instructions have been processed in the sampling period by multiplying the CPU time (first three fields in /proc/stat file) and the core frequency. The result could be used as the response time metrics.

DVFS Interface

We enable the Intel's SpeedStep on BIOS and use cpufreq package to implement DVFS. When using root privilege, we can echo different frequencies into the system file `/sys/devices/system/cpu/cpu[X]/cpufreq-scaling_setspeed`, where [X] is the index of the core number. We test the overhead of scaling CPU frequencies on our platform, which is only 0.08 milliseconds on average.

Power Estimation

We measure the processor's power consumption by connecting two multi-meters into the circuit, as shown in Figure 3.10. Specifically, as the processor is connected by two +12V CPU 4 pin cables, we put one multi-meter in each cable to test the amperage (A). On the other side, the

Agilgent IntuiLink software logged data into the logging server using Microsoft Excel. After we collecting the measured amperage based on the sampling period, we are able to compute the average amperage accordingly, such that we could obtain the energy by multiplying the voltage, amperage as well as the time duration.

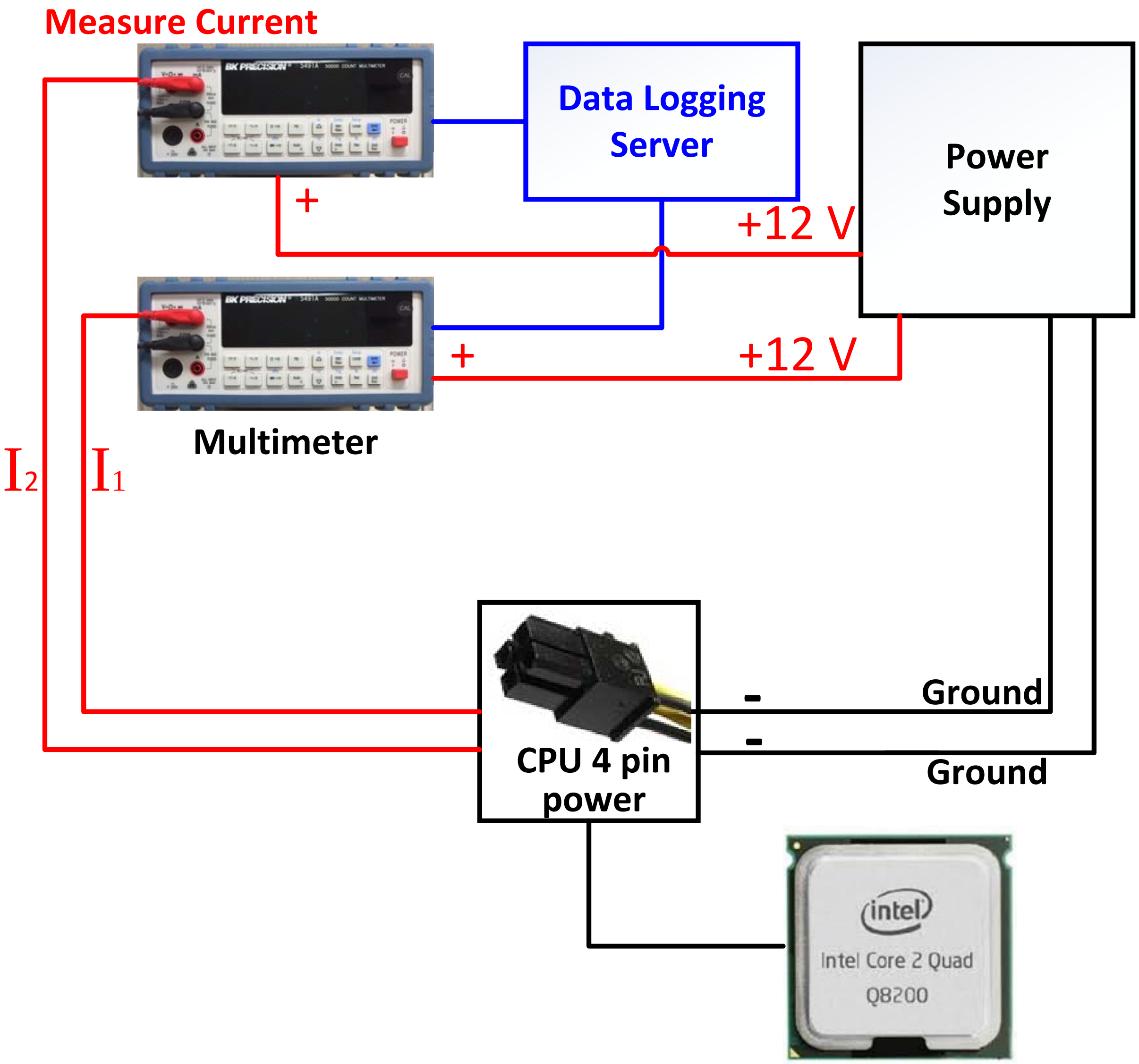


Figure 3.10: The Measurement setup: two multimeters are inserted between the power supply and CPU

Baseline Control Methods

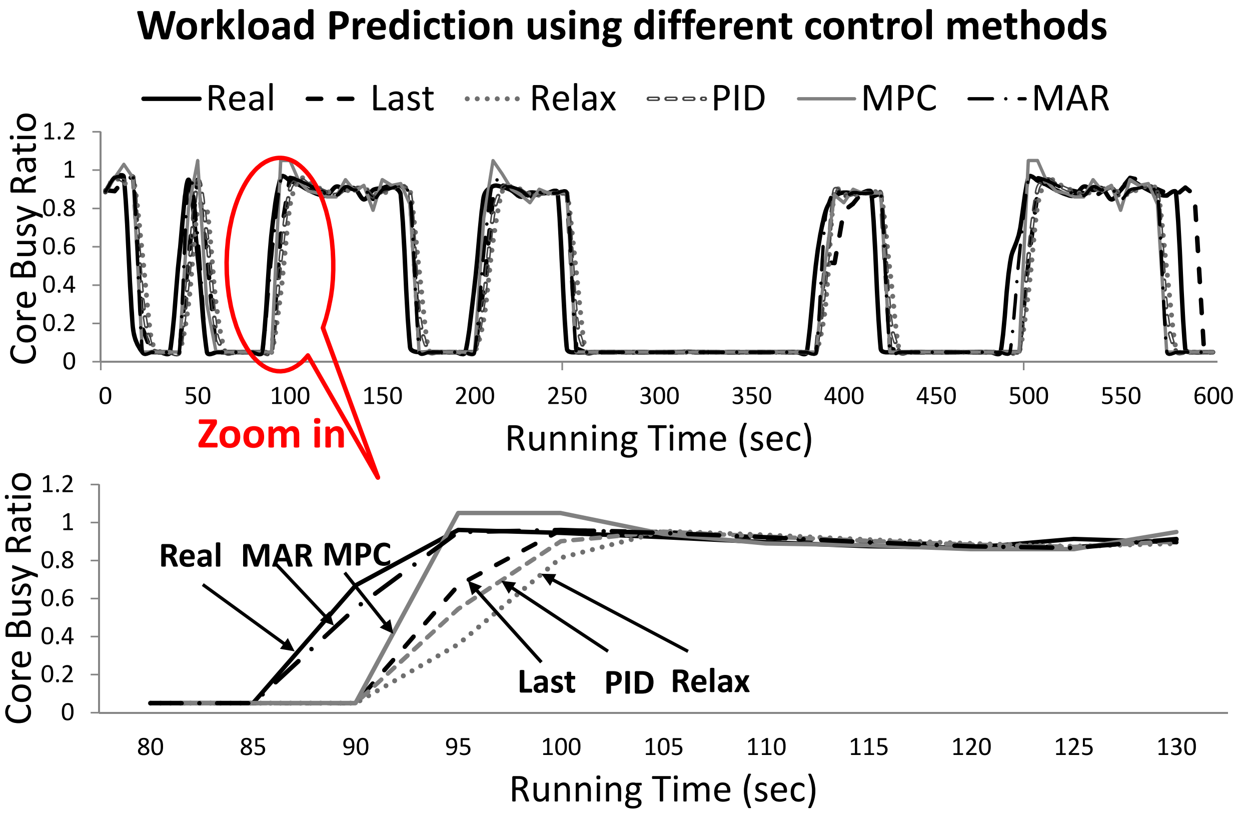
PID controller [32] is a control loop feedback mechanism widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired setpoint. Model predictive control (MPC) [33] is an advanced method of process control that has been widely used in the process industries. It relies on dynamic models of the process, most often linear empirical models obtained by system identification. LAST [9] is the simplest statistical predictor, which assumes the next sample behavior is identical to its last seen behavior. Relax [25] is an algorithm which predicts the workload using both history values and run-time profiling.

Experiments

First, MAR is compared with four other baselines to illustrate the high responsiveness of MAR. Second, MAR is used to do the power control for different types of workloads, including CPU-intensive, memory-intensive and I/O-intensive benchmarks. The purpose is to show MAR's performance under specific environment. Third, we compare the two versions of MAR (with/without considering I/O wait) by running data-intensive benchmarks, in order to highlight the impact of I/O wait in power management schemes. After that, we compare the overall efficiency of MAR and the baselines. In order to demonstrate MAR's efficiency in power management, we also compare MAR with conventional Linux governors solution (Ondemand). At the end, we briefly evaluate the overhead of our power management schemes.

Fast Responsiveness

In this section, we compare MAR's response time as well as prediction accuracy with four baselines: LAST [9], PID [32], Relax [25] and MPC [33]. All of these algorithms are implemented in RTAI4.0 [34] to trace the cores' behavior and predict the next core-boundness. LAST is the simplest statistical predictor, which assumes the next sample behavior is identical to its last seen behavior. For RELAX, we set the relaxation factor to 0.5 based on the empirical value in [25]. For PID, we tune K_P=0.4,K_i=0.2,K_d=0.4 based on [32]. In MPC implementation, the prediction horizon size is 2, the control horizon size is 1 as described in [35], and the setpoint is set to the average core boundness obtained from offline computation. First, we use Figure 3.11 to show the trajectory of prediction when running bzip2 benchmark with SP=10s.We also run gcc, mcf, gap, applu, gzip, and collect the "average detecting times" for all of the benchmarks, as shown in the table in Figure 3.11 . It can be seen that MAR has the fastest response time (as shown in the zoomed area in Figure 3.11). The reason that MAR response quickly than other methods because it build up based on fuzzy logic which is a solution created by the knowledge of the system. The more we understand about the system, the less uncertainty would be appeared. The table in Figure 3.11 also proves that MAR achieves the shortest settling time after the deviation. The overall response time of MAR outperforms LAST, Relax, PID, MPC by 56.5%, 104%, 174% and 43.4% respectively. Second, we measure the impact of SP in prediction accuracy. Figure 3.12 shows the average prediction errors for all five algorithms when SP=5s, SP=10s and SP=20s. We can see that all predictors perform better when SP=10s. When using a smaller SP, MAR could respond to the changes of core boundness more quickly but may lead to some possible over-reactions; when using a larger SP, the core's behavior is going to change slower due to the larger time window but will be more stable. Slow responsive algorithms such as PID do not work well here since they are only good for the workloads with strong locality. In summary, MAR always obtains the least prediction error because it incorporates the tracking error, which gives more hints for the coming trend and high responsiveness for core's status switches.



Average time	LAST	RELAX	PID	MPC	MAR
Detecting the bouncing	1.8	2.35	3.15	1.65	1.15
Setting after bouncing	3.25	3.85	5.25	3.15	1.25
Average Response time	5.35	6.25	8.5	5	2.25

Figure 3.11: Comparison of workload prediction accuracy on a randomly picked core, SP=10s.

Power Efficiency

This set of experiments shows the power management efficiency of MAR for different types of benchmarks: gcc, mcf, bzip2, gap, applu, gzip and TPC-C.

Running homogeneous workloads: In this section, we want to show MAR's control performance when homogeneous workloads are running. For each benchmark, we use 4 threads to run its 4 copies on our testbed to evaluate the MAR's performance for each specific type of workloads. We show the results of power consumption/performance loss of MAR and the baselines: LAST, Relax, PID, MPC and the Ideal case in Figure 3.13. In "Ideal" case, we use the ideal DVFS settings calculated offline, which could achieve the best power saving efficiency and the least performance loss. Assuming the ideal case saves the most power, MAR and other base-lines perform well when the workloads have no explicit I/O operations. For gcc, mcf, bzip2, gap, applu and gzip, MAR is 95.4% close to the ideal power management, while LAST is 91.7%, Relax is 94.1%, PID is 93.6%, MPC is 94.5%. However, when we run TPC-C benchmark, the baselines can only achieve 57.8%-69.8% power saving performance as the ideal case. With considering I/O wait as opportunities to save power, MAR can still achieve 92.5% of the power efficiency of Ideal case. At the same time, the performance loss of all power management strategies is between 2%-3%. Although MAR has the highest performance loss 2.9% for TPC-C benchmark (because of our aggressive power saving strategy), it is still in the safe zone [32].

Running heterogeneous workloads: This section is to compare MAR with the baselines when heterogeneous workloads are running; we still launch all afore-mentioned 7 benchmarks in parallel on our testbed. The database for TPC-C benchmark is locally set up. Figure 3.14 shows their overall DVFS results and power-saving efficiency. The upper two charts in Figure 3.14 illustrate the frequency distributions of all management methods. Note that compared with SP=5s, the trajectory of workload in SP=10s case has less fluctuations caused by the "phantom bursts". The slow-responsive methods such as Relax, PID and MPC could not discover as many power-saving opportunities as the fast-responsive ones: MAR and Last, especially in the smaller SP case. From the upper left of Figure 3.14, we can see that 60% of MARs DVFS result is running under the lowest frequency, nearly 10% is set to the medium level. MAR completely outperforms the other four different DVFS control solutions.

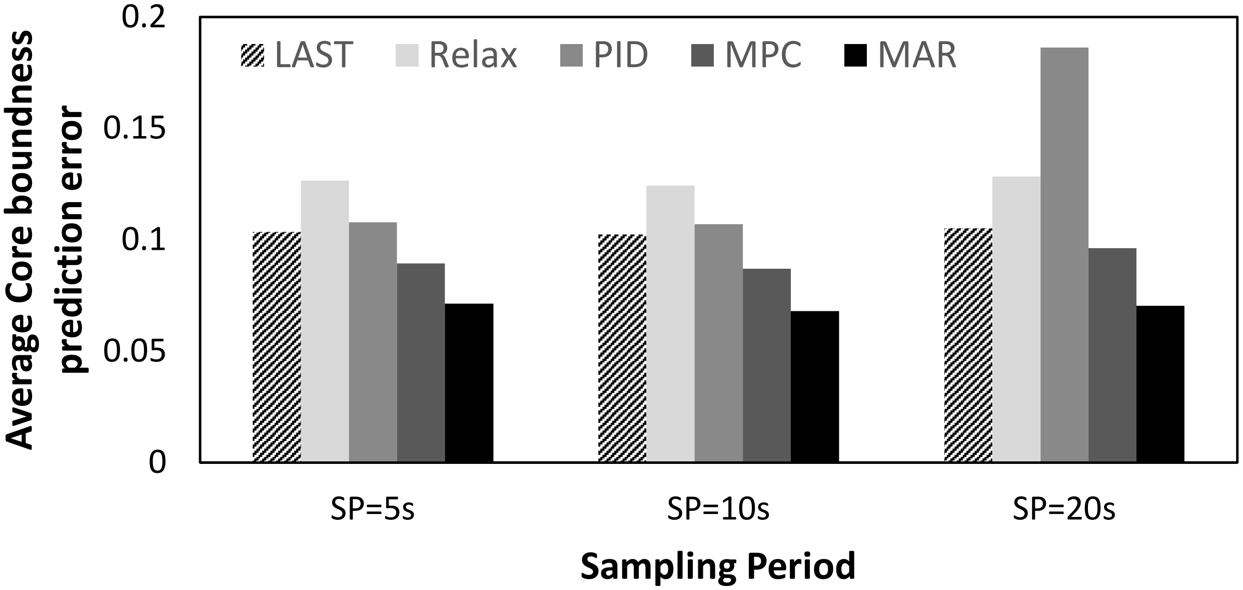


Figure 3.12: Average prediction errors for different sampling period

The lower two charts in Figure 3.14 describe the power consumption of all management methods. All the numbers are normalized to MAR which saves the most power. PID and MPC perform very differently when SP=10s and SP=5s. The reason is that more "phantom bursts" of the workloads (when SP=5s) could affect the control accuracy significantly. LAST is always better than Relax because it is more fast-responsive to the core's status switches in CMP environment. From the power saving perspective, MAR, on average (SP=10/5s), saves 5.6% more power than LAST, 3.7% more than Re-lax,2.9% more than PID, 2.1% more than MPC.

The impact of the new model B-W-I: In order to highlight the impact of B-W-I model in power management, we conducting the experiments on two different MARs as we described in Section 3 which are MAR and MAR(B-W-I Model). The former one is used to test the fast-responsive to the status switches but does not consider I/O effects. We use 7 threads to run gcc, mcf, bzip2, gap, applu, gzip and TPC-C in parallel. The comparison of MAR and MAR (B-W-I Model)is shown in Figure 3.15. The results show that MAR (B-W-I Model) is more likely to use lower frequencies than MAR.

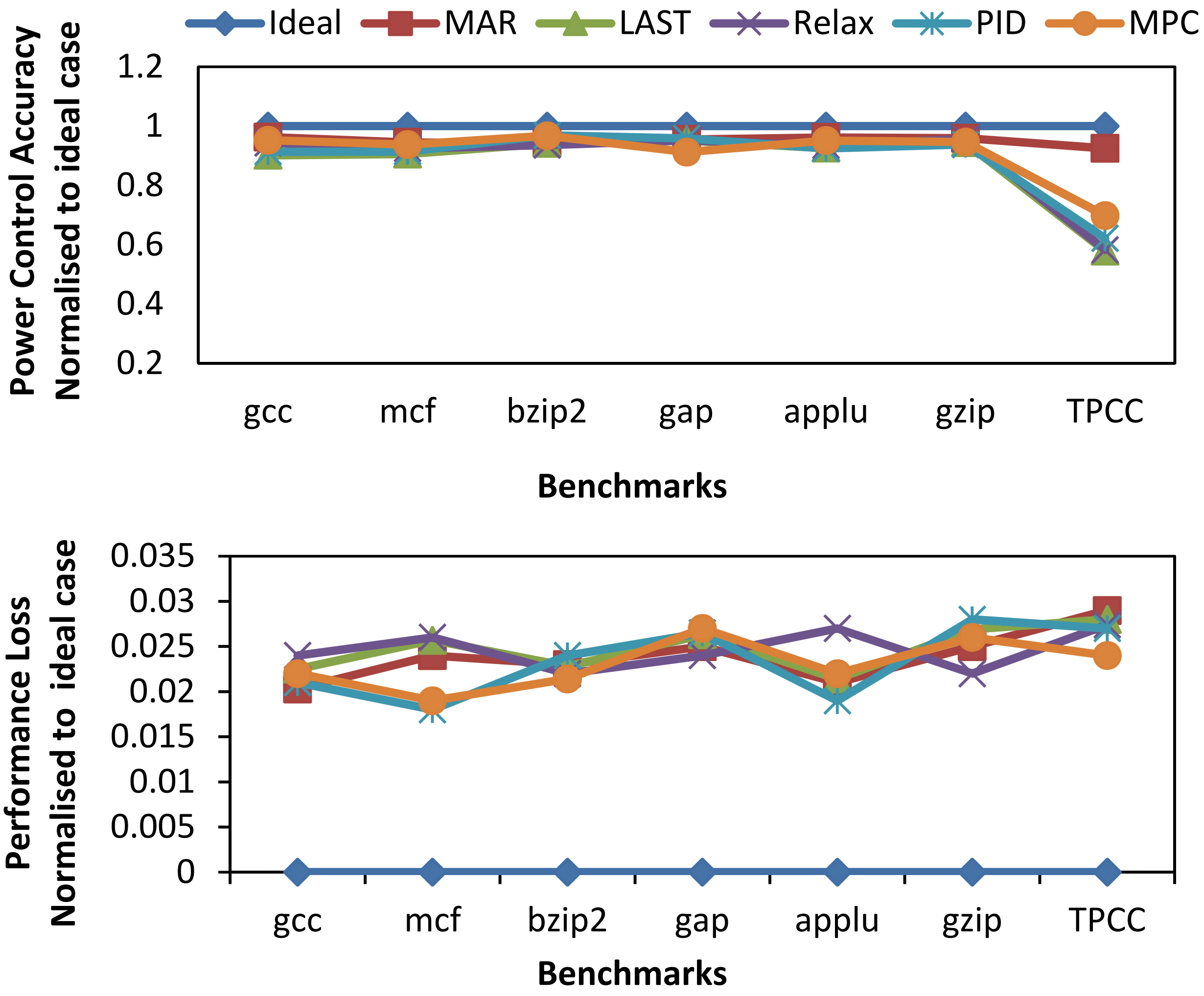


Figure 3.13: MAR's performance comparison under various benchmarks

The reason is: when the I/O wait exceeds the thresholds in the control period, even if the response time is close to RRT MAR (B-W-I Model) still scales down the core frequency to a lower level to save more power. Compared with MAR (B-W-I Model), MAR cannot discover the potential I/O jobs which are overlapped with the computing intensive jobs. Based on the cubic relation between frequency and power consumption, when SP=10s, MAR (B-W-I Model) could save 10% more power than MAR; when SP=5s, MAR (B-W-I Model) saves 9.7% more power.

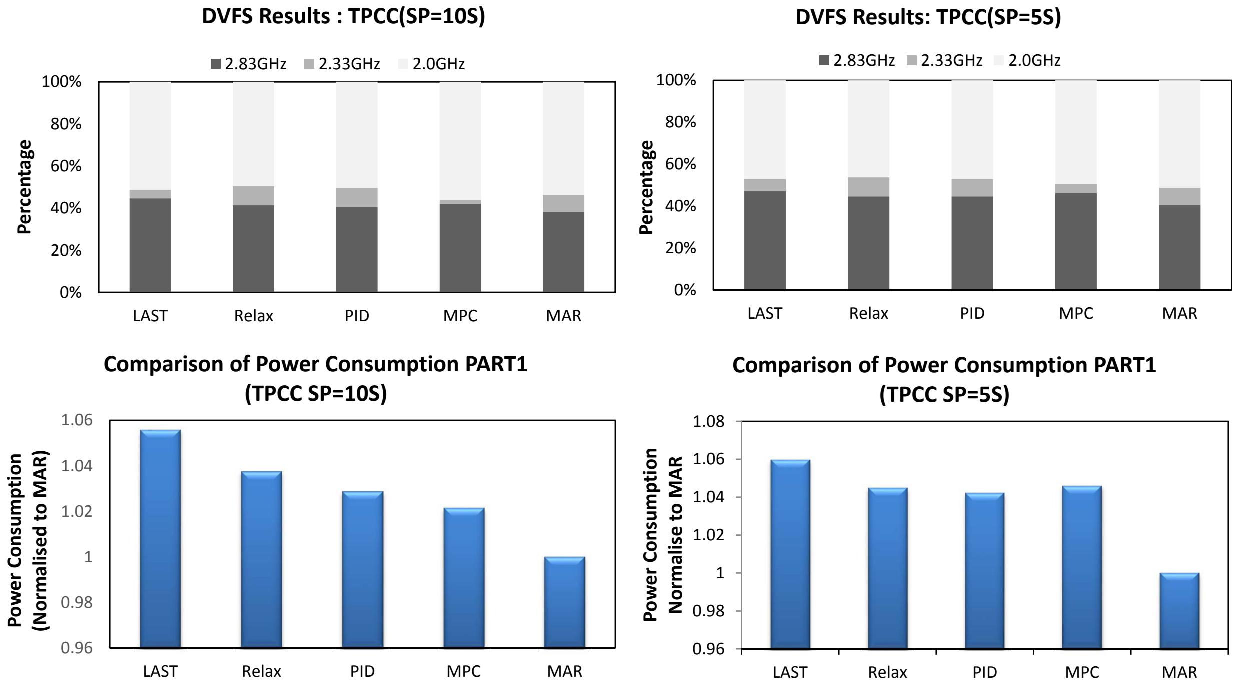


Figure 3.14: Comparison of the power management efficiency of MAR with the baselines, SP = 10s/5s

We plot the power consumption and performance statistics of MAR, MAR (B-W-I Model), the performance oriented case, as well as the ideal case in Figure 3.16. In "Perf.Oriented" case, maximum frequencies are used all the time. All the numbers are normalized to "Perf. Oriented" which has the least power consumption. Based on the results, MAR (B-W-I Model) saves about 9.7%-10% more power than MAR on average. It is expected that the more I/O intensive the workloads are, the better performance the MAR (B-W-I Model) could achieve.

Overhead: At the end, Table IV shows the overhead of the tested methods. They are all lightweight and consume less than 1% CPU utilization for sampling period of 10s. The MPC controller has the highest overhead because it is computationally expensive. MAR executes almost 9 times faster

Table 3.4: Comparison of the Overhead of Different Managements

	MAR	**LAST**	**Relax**	**PID**	**MPC**
Code size (Lines)	160	50	50	135	600
CPU utilization	0.12%	0.05%	0.05%	0.09%	0.92%

than MPC controller.

Scalability: In previous subsections, we have tested MAR on our testbed, which only has 4 and 3 available voltage-frequency settings. In order to show the scalability of MAR, we use cycle-accurate SESC simulator with modifications to support per-core level DVFS. Each core is configured as Alpha 21264 [37]. We enable Wattchify and cacify [36] to estimate the power change caused by DVFS scaling. In our simulation, we scale up MAR for 8, 16, 32 core processors with private L1 and L2 hierarchy and the cores are placed in the middle of the die. Each core in our simulation has 3 DVFS levels (3.88 GHz, 4.5 GHz and 5 GHz). The over-head of each DVFS scaling is set to 20 [3]. The bench-marks we used are randomly selected SPEC 2006 benchmarks: gcc, mcf, bzip2 and data-intensive TPC-C benchmark. The number of processes equals to the number for cores, e.g. we run 2 copies of each of the 4 benchmark when there is 8 cores. We first record the maximum power consumption and the best performance of the workloads by setting all the cores at the highest DVFS level. Then we normalize the results of MAR and other baselines to show their power management efficiency and performance loss. Figure 3.17 plots the average power saving efficiency and the performance loss of MAR and LAST, RELAX, PID, MPC based per-core level DVFS controllers. All the numbers are normalized to the "Performance-Oriented" case. With different number of cores, the CMP processor under MARs monitor always saves the most power: about 65% compared with cases without DVFS control. On average, MAR outperforms LAST, Relax, PID and MPC about 14%, 12.3%, 11.8% and 10.1%, respectively, under our benchmark configurations. At the same time, MAR and the baselines performance loss are all be-tween 2%-

3%, which confirm what we have observed on our testbed. Our simulation results demonstrate that MAR can precisely and stably control power to achieve the performance requirement for CMPs with different number of cores.

Power conservation potential for multiple CPU frequency levels: In order to further investigate the relationship between MAR's power conservation potential and multi CPU frequency levels, we conducted more experiment in a new testbed with five frequency levels. We launched the data-intensive benchmark (TPC-C) in parallel on our testbed and recorded the DVFS control outputs for MAR, Last, Relax, PID, and MPC. The detailed results are illustrated in Figure 3.18. We calculate the power consumption based on the frequency cubic function that are documented in Section 3 and also present the results in Figure 3.18. The upper chart in Figure 3.18 shows the DVFS results for running TPC-C benchmark while the sampling period equals to 5s. From the figure, we can see that MAR is able to scale the CPU frequency at the lowest level for nearly 40% of the entire running period, while the corresponding results for Last, Relax, PID, MPC are 23%, 19%,20% and 21% respectively. The lower chart compares the power consumption among all the five control methods with respect to MAR. It is clear that MAR can save the most power, which is about 20%, 24%, 21%, 16% more than Last, Relax, PID and MPC respectively. This further demonstrates MAR's better potential in energy conservation when the processors support more CPU frequency levels.

Comparison with conventional governors

In this section, we compare the power conservation capability between our MAR and the conventional Linux governors "Ondemand". Power management techniques of Linux is conducted in Linux kernel-level, where the frequency scaling policies are defined in the form of frequency scaling governors [45]. CPU adapts the frequencies based on workload through a user feedback mechanism. As a result, if CPU has been set to a lower frequency, the power consumption will be reduced.

The CPU freq infrastructure of Linux allows the CPU frequency scaling handled by governors. These governors can adjust the CPU frequency based on different criteria such as CPU usage. There are basically four governors in the kernel-level power management scheme, which are “Performance”, “Powersave”, “Ondemand” and “Conservative”. Specifically, the Ondemand governor can provide the best compromise between heat emission, power consumption, performance and manageability. When the system is only busy at specific times of the day, the “Ondemand” governor will automatically switch between maximum and minimum frequency depending on the real workload without any further intervention. This is a dynamic governor that allows the system to achieve maximum performance if the workload is high and scale the frequency down to save power when the system is idle. The “Ondemand” governor uses traditional B-I model that we introduced in Section 3 for frequency scaling. It is able to switch frequency quickly with the penalty in longer clock frequency latency. The “Performance” governor forces the CPU to set the frequency at the highest level to achieve the best performance. In this case, it will consume the highest power. The “Powersave” governor will conserve the most power with a lowest system performance.

In order to demonstrate MAR’s effectiveness and efficiency in power conservation as compared with Linux conventional governors, we run the data intensive benchmark TPC-C in our testbed and set the Linux governor to “Ondemand” status. In the meantime, we record the DVFS results for MAR, MAR(BWI), and “Ondemand” respectively. Figure 3.19 illustrates the comparison results of the above-mentioned three methods. Based on the results, we can see that both MAR and MAR(BWI)outperforms the conventional governor “Ondemand” up to 8.62% and 17% in power conservation.

Conclusion

Power control for multi-core systems has become increasingly important and challenging. However, existing power control solutions cannot be directly applied into CMP systems because of the new data-intensive applications and complicate job scheduling strategies in CMP systems. In this work, we present MAR, a model-free, adaptive, rule-based power management scheme in multi-core systems to manage the power consumption while maintain the required performance. "Model-less" reduces the complexity of system modeling as well as the risk of design errors caused by statistical inaccuracies or inappropriate approximations. "Adaptive" allows MAR to adjust the control methods based on the real-time system behaviors. The rules in MAR are derived from experimental observations and operators' experience, which create a more accurate and practical way to describe the system behavior. "Rule-based" architecture also reduces the development cycle and control overhead, simplifies design complexity. MAR controller is highly responsive (including short detective time and settling time) to the workload bouncing by incorporating more comprehensive control references (e.g. changing speed, I/O wait). Empirical results on a physical testbed show that our control method could achieve more precise power control result as well as higher power efficiency for optimized system performance compared to other four existing solutions. Based on our comprehensive experiments, MAR could outperform the baseline methods by 12.3%-16.1% in power saving efficiency, and maintains comparable performance loss about 0.78%-1.08%. In our future research, we will consider applying fuzzy logic control in power conservation of storage systems.

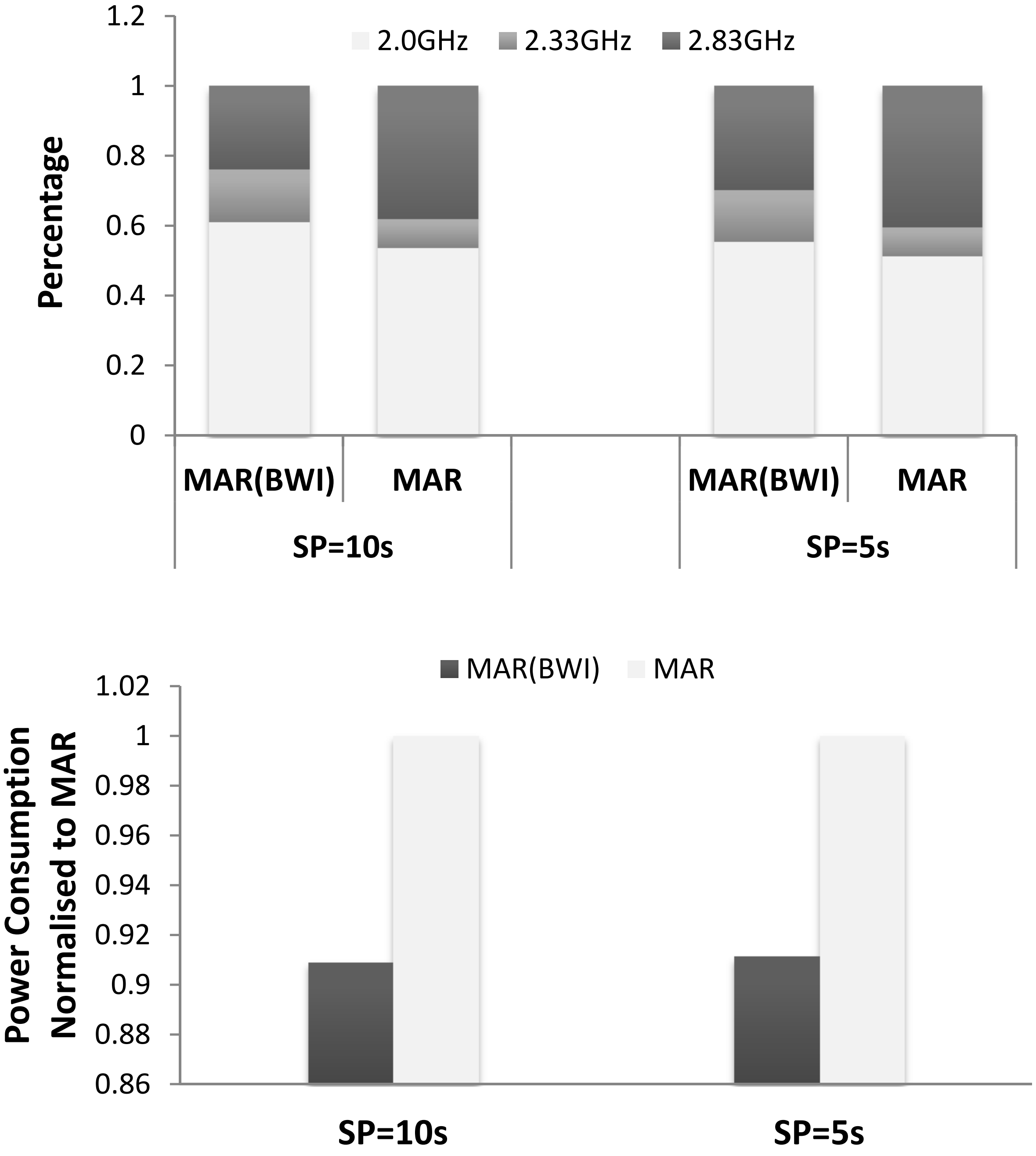


Figure 3.15: The DVFS and power consumption results of MAR (BWI) and MAR

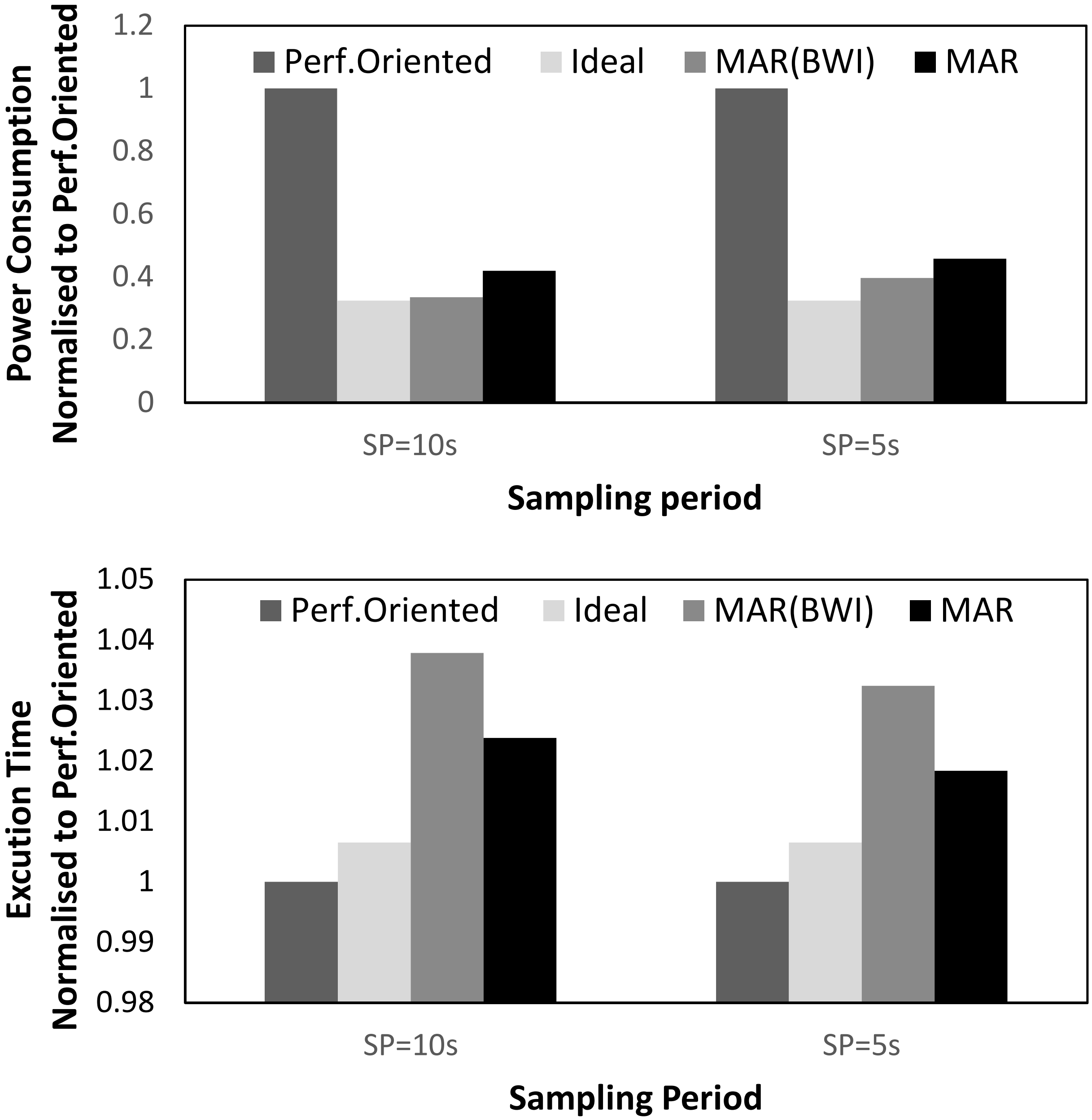


Figure 3.16: Comparison of Power-saving efficiency and performance loss, SP = 10s/5s

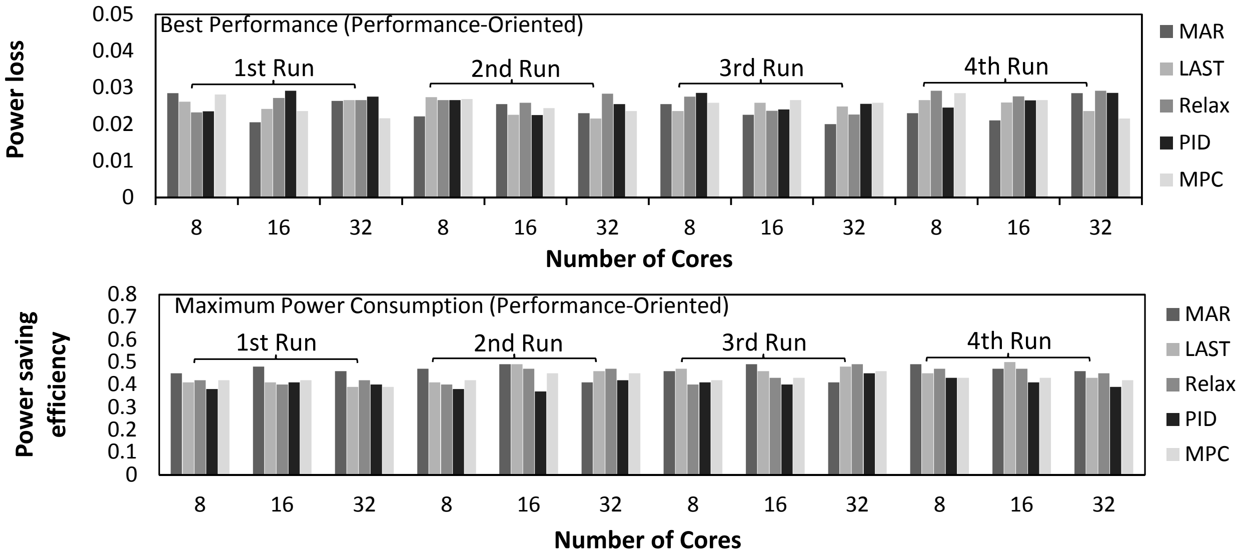


Figure 3.17: Scalability study of MAR and baselines under different number of cores in simulation

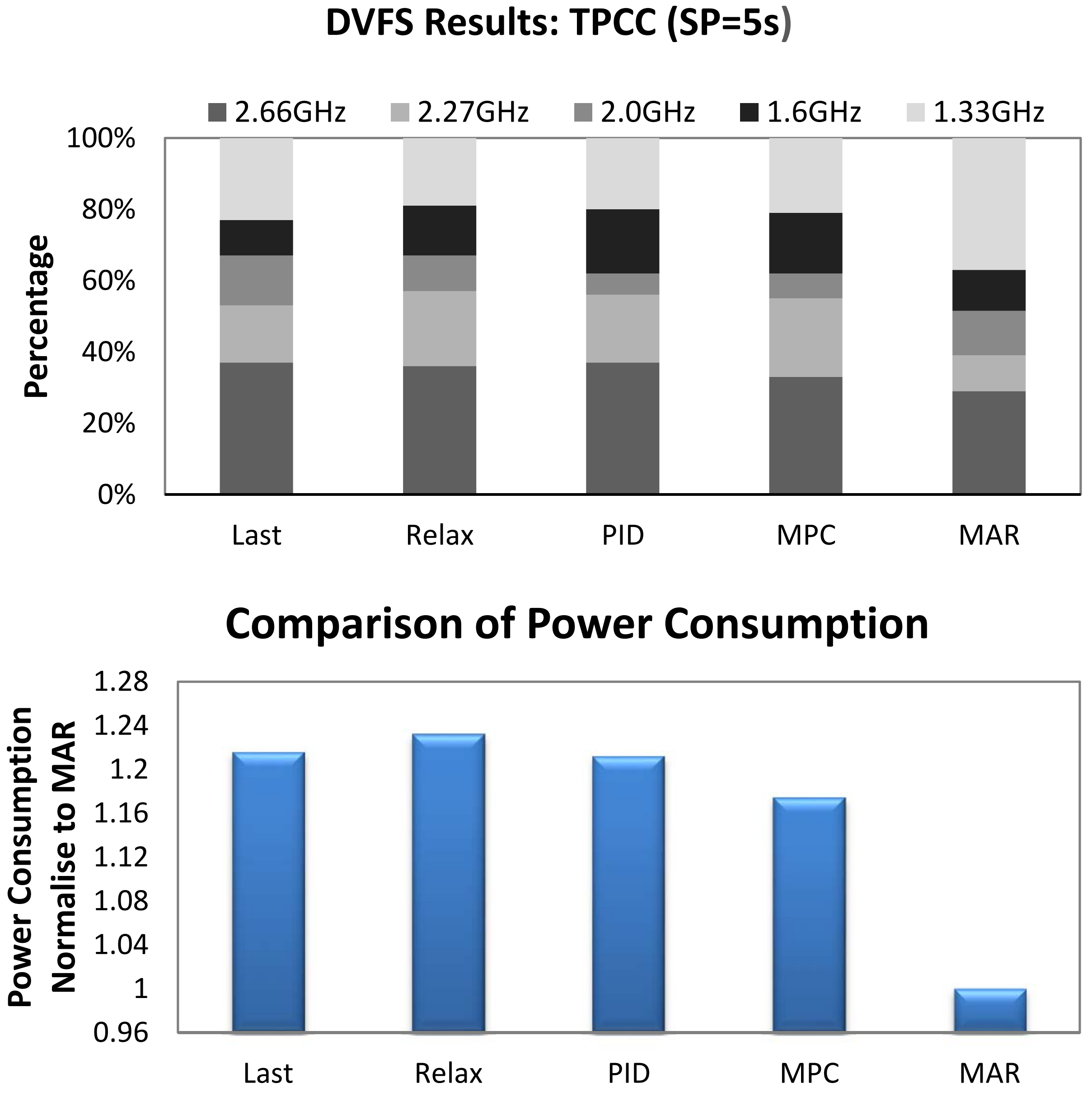


Figure 3.18: Comparison of DVFS Scaling results for MAR and baselines with 5 CPU frequency levels

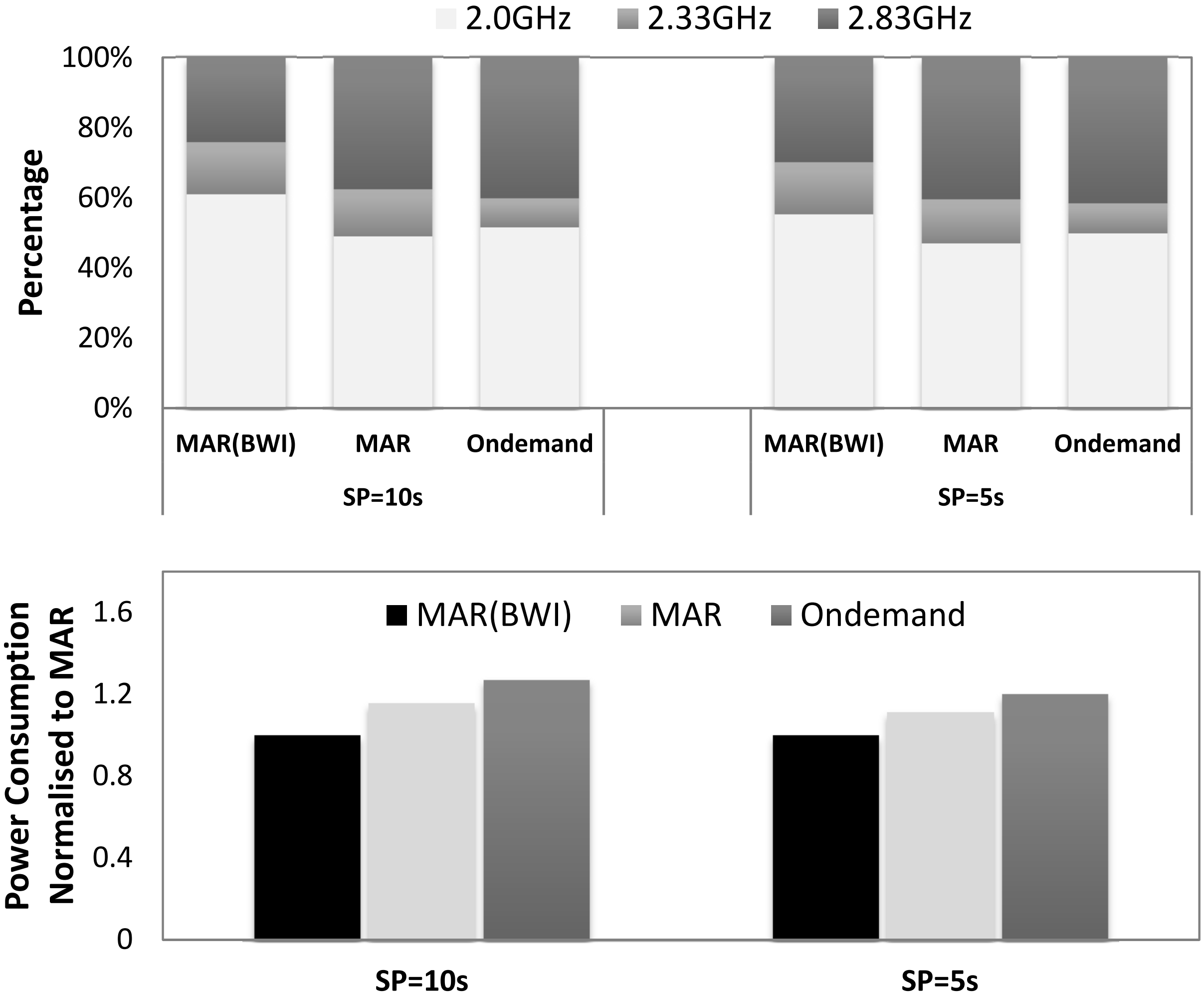


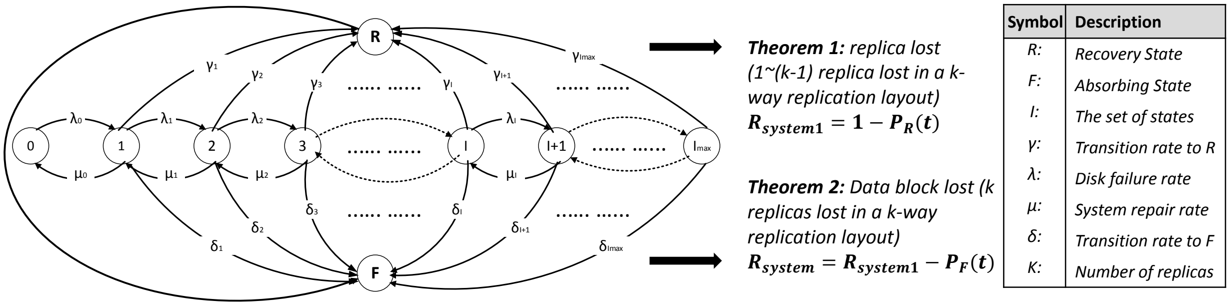
Figure 3.19: Performance and Power Consumption Comparison between MAR and Linux conventional governors in DVFS

CHAPTER 4: A NEW RELIABILITY MODEL IN REPLICATION-BASED BIG DATA STORAGE SYSTEMS

Extended Reliability Model

Continuous time Markov chain (CTMC) has been utilized for the modeling of multi-way replication declucstered storage system [81]. In this section, we will further extend the model by abstracting the CTMC model to an ordinary differentiate equation (ODE) group. In order to precisely analyze the reliablity of the storage system, we made the following assumptions:

- The state space I is the set of states with a certain number of failed disks.
- A disk failure means that the data of the failed disk is unreachable and unrecoverable in place permanently. Other types of failures such as problems in cables, network, air conditioning and power are not considered.
- All the disks are homogeneous with the same failure rate.
- Storage systems with different layouts store the same amount of data per disk.



Theorem 1: *replica lost (1~(k-1) replica lost in a k-way replication layout)*
$\boldsymbol{R_{system1} = 1 - P_R(t)}$

Theorem 2: *Data block lost (k replicas lost in a k-way replication layout)*
$\boldsymbol{R_{system} = R_{system1} - P_F(t)}$

Symbol	Description
R:	*Recovery State*
F:	*Absorbing State*
I:	*The set of states*
γ:	*Transition rate to R*
λ:	*Disk failure rate*
μ:	*System repair rate*
δ:	*Transition rate to F*
K:	*Number of replicas*

Figure 4.1: State transition in declustered layouts

- Recovery procedure starts right after disk failure.
- The recovery is at block level, and the switch overhead (seeking and rotating time) between normal service and recovery is not considered.
- The storage network bandwidth is not a bottleneck in parallel recovery.
- The workload per disk in the storage system follows an even distribution. The storage system is not saturated at any time point.

With these assumptions, CTMC is the desired model to describe the development of a storage system. Define $p_i(t)$ as the probability that the system is in the state i at time t. The graphical representation of a CTMC was presented in Figure 4.1 and can be abstracted to an ordinary differentiate equation (ODE) group shown in equation 4.1. With regards to the system reliability, there are two situations we should taking into consideration. Take 3-way replication method as an example, the first situation happens when one or two replica of the data blocks have been lost due to node failures,disk failures or system unrecoverable errors. In this case, the data lost instance was not happened but the system reliability was degraded because of only one or two copies of the data block remaining functionally. The second situation happens when the data lost instance was happened, which means 3 replica of the data blocks have been lost. In this case, the data blocks will lost permanently, which should be avoided in the storage system design. We will analysis and formulate the above two cases in details in Sections 4 and 4.

$$\frac{d}{dt}\mathbf{P} = Q\mathbf{P} \tag{4.1}$$

where Q is the generator matrix [70], $\mathbf{P}$ is defined $\mathbf{P} = [p_0(t), p_1(t), \ldots, p_R(t), p_F(t)]'$ as the vector of $p_i(t)$'s, and $\dfrac{d}{dt}\mathbf{P}$ is defined $\dfrac{d}{dt}\mathbf{P} = [\dfrac{d}{dt}p_0(t), \dfrac{d}{dt}p_1(t), \ldots, \dfrac{d}{dt}p_R(t), \dfrac{d}{dt}p_F(t)]'$ as the vector of the first derivative of $p_i(t)$'s for all i.

The element of Q on the i-th ($0 \leq i \leq l_{max}+1$) row and the j-th ($0 \leq j \leq l_{max}+1$) column (represented by q_{ij}) is the transition rate from the state i to the state j. The elements of the generator matrix of the CTMC in are:

$$q_{ij} = \begin{cases} \lambda_i, \text{ if } 0 \leq i \leq l_{max-1} \text{ and } j = i+1 \\ \mu_i, \text{ if } 1 \leq i \leq l_{max} \text{ and } j = i-1 \\ \delta_i, \text{ if } 1 \leq i \leq l_{max} \text{ and } j = F \\ \gamma_i, \text{ if } 1 \leq m \leq k-1, \ 0 \leq i \leq l_{max-1} \text{ and } j = R \\ -\sum_{i \neq j} q_{ij}, \text{ if } i = j \end{cases} \tag{4.2}$$

Equation Group (4.1) can be expanded in terms of Equation Group (4.3) by q_{ij}'s into it.

$$\begin{cases} \dfrac{dp_0(t)}{dt} = -\lambda_0 p_0(t) + \mu_0 p_1(t) \\ \dfrac{dp_i(t)}{dt} = -(\mu_{i-1} + \lambda_i + \delta_i) p_i(t) + \lambda_{i-1} p_{i-1}(t) + \mu_i p_{i+1}(t), \\ \qquad 1 \leq i \leq l_{max} \\ \dfrac{dp_R(t)}{dt} = \displaystyle\sum_{i=0}^{l_{max-1}} \sum_{j=1}^{k-1} \gamma_i p_{b_j}(t) \\ \dfrac{dp_F(t)}{dt} = \displaystyle\sum_{j=1}^{l_{max}} \delta_j p_j(t) \end{cases} \tag{4.3}$$

The functions $p_i(t)$'s ($0 \leq i \leq F$) solved from the ODEs (4.3) with the initial condition: $p_0(0) = 1$, $p_i(0) = 0$ ($\forall i > 0$) is the probability that the system is in state i at time t. The initial condition means at the initial time, the probability of the system without disks failure (at state 0) is 100%.

Case 1:

In this section, we will analyze and formulate the relationship between system reliability and the probability of replica lost. The state R represents the recovery state, which means 1 to $k-1$ replicas have been lost in a k-way replication approach. The system reliability has been degraded and was defined in equation (4.4):

$$R_{system1}(t) = 1 - p_R(t) \tag{4.4}$$

R represents the recovery state, the function $p_R(t)$ is defined as the system unreliability under recovery mode. We will discuss the aggressive parallel recovery in Section 4. As we discussed in the above paragraph, $p_R(t)$ is closely related with the probability of replica lost p_b. Now, we further investigated and computed the probability of replica lost defined in equation (4.5).

$$p_b(m) = \begin{cases} \dfrac{m}{C(N,1) \cdot C(k,m)}, \\ \text{m: } 1 \leq m \leq k-1, \end{cases} \tag{4.5}$$

where m stands for the number of replicas, k defined as the total number of replicas for each data block, and N represents the total number of data blocks per disk. Based on the principle of permutation, the equation (4.5) can be further expanded and shown in the following equation (4.6).

$$p_b(m) = \begin{cases} \dfrac{m \cdot m!}{N \cdot k \cdot (k-1) \cdot \ldots \cdot (k-m+1)}, \\ \text{m: } 1 \leq m \leq k-1, \end{cases} \tag{4.6}$$

Apparently, $p_b = 0$ if $m = 0$. If the replica lost after one more disk failure, the system transits from the state l to state R, and the transition rate γ_l was defined as:

$$\gamma_l = (n - l) \cdot \lambda \cdot (1 - p_b) \tag{4.7}$$

The transition rate γ_l is related to the number of failed disks, the disk failure rate and the probability of replica lost. In the next section, we will investigate the other situation that will affect the system reliability mentioned above.

Case 2:

Since the state F represents the absorbing state, the function $p_F(t)$ is usually defined as the system unreliability [75]. Meanwhile, the reliability of the system is defined in Equation (4.8):

$$R_{system}(t) = R_{system1}(t) - p_F(t) \tag{4.8}$$

The mean time to failure (MTTF) of the system described by an absorbing CTMC is defined in Equation (4.9) [75]. In this particular model for a storage system, F is the state of losing data, and the MTTF of the system is actually the mean time to data loss (MTTDL).

$$MTTDL = \int_0^\infty R_{system}(t)dt \tag{4.9}$$

In our model, λ_l and δ_l are functions of $P^{(l)}$, which is defined as **the probability that the system does not lose data after the l-th disk failure**. Apparently, $P^{(l)} = 1$ if $l < k$, and $P^{(l)} = 0$ if $l > l_{max}$. With l failed disks, there are $n - l$ disks which may fail, so the rate of another disk failure is $(n-l)\lambda$, where λ represents the disk failure rate. If there is no data loss upon l disk failures, and no data loss after one more disk failure, the system transits from the state l to $l+1$, and the transition rate is $(n-l)\lambda P^{(l+1)}$. On the other hand, if there is data loss after one more disk failure, the system transits from the state l to the state F, and the transition rate is $(n-l)\lambda(1-P^{(l+1)})$. Accordingly,

$$\lambda_l = (n-l)\cdot\lambda\cdot P^{(l+1)} \tag{4.10}$$

$$\delta_l = (n-l)\cdot\lambda\cdot(1-P^{(l+1)}) \tag{4.11}$$

The value of $P^{(l)}$ and the repair rate μ_l when the system has l failed disks are also different from layout to layout. In the following section, we will investigate the above two values in our shifted declustering layout as well as random declustering layout in the later sections.

Reliability Analysis

Shifted Declustering Layout

Shifted declustering [80] is a layout scheme that distributes the replicas within one redundancy group with a certain distance. The distance increases in each iteration by one. Depending on the number of disks n, shifted declustering has two solutions:

1. Basic solution

   The basic solution is applied to the situations when n is odd or $n=4$ for $k=3$, and n is prime for $k \geq 4$.

   For k-way replication ($k \geq 3$), the k replicas within one redundancy group are distributed to k disks, and from the first disk in these k disks on, each disk has a fixed distance after the previous one.

For example, assuming that the disks are labeled from 0 to $n-1$, if $k = 3$, and one redundancy group is distributed to disks d_1, d_2 and d_3, then $d_2 = (d_1 + y) \bmod n$, and $d_3 = (d_2 + y) \bmod n$, where y is the distance between in replicas of this redundancy group. The layout is illustrated in Figure 4.2 and explained in [80].

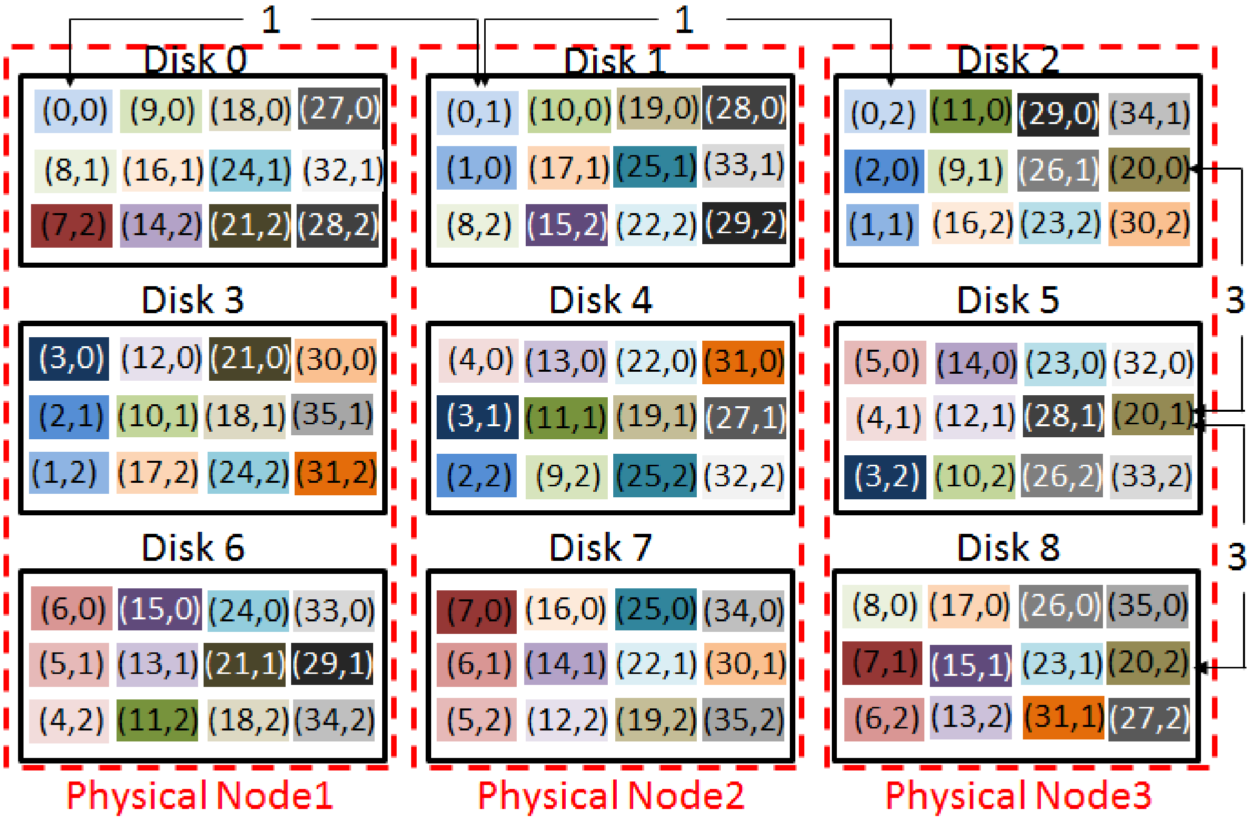


Figure 4.2: Shifted declustering layout

Due to the layout feature, data will be lost upon l disk failures if there exists a set of k failed disks with a fixed distance between neighboring disks. Otherwise, there is no data loss. **The value of $P^{(l)}$ (no data loss after l disk failures) is the probability that given l failed disks, for any k out of the l disks, there is no fixed distance between neighboring ones.**

Therefore, we are looking for the sub-suite $\{i_0,\ i_1,\ \ldots,\ i_{l-1}\} \subseteq [0,\ \ldots,\ n-1]$ such that $\forall\{j_0,\ j_1,\ \ldots,\ j_{k-1}\} \subseteq \{i_0,\ i_1,\ \ldots,\ i_{l-1}\}$, where $\{j_0,\ j_1,\ \ldots,\ j_{k-1}\}$ is a combination of k elements out of the sub-suite $\{i_0,\ i_1,\ \ldots,\ i_{l-1}\}$, and not exist d for all $m \in \{0,\ 1,\ \ldots,\ k-1\}$, $j_{(m+1) \bmod k} = (j_m + d) \bmod n$.

Denote $S_{sd}(n,k,l)$ as the number of choices to select l disks from n disks with k-way replication configuration, obeying the conditions shown above. Therefore,

$$P^{(l)} = \frac{S_{sd}(n,k,l)}{C(n,l)} \tag{4.12}$$

In particular, if $l = k$, $P^{(k)} = 1 - n(n-1)/2C(n,k)$, because if first failed disk is chosen, there are $(n-1)/2$ types of distances to decide the other $k-1$ failed disks to cause data loss. There are n ways to select the first disk, so there are in total $n(n-1)/2$ ways to select k failed disks to lose data.

2. Extended solution

The extended solutions are applicable to the cases when n is even and $n > 4$ for $k = 3$, and n is non-prime for $k \geq 4$.

For the extended, replace n in the numerator in Equation 4.12 with n', where n' is the maximum number that is smaller than n and there exists a basic solution for n'-disk, k-way replication configuration. If $k = 3$, $n > 4$ and n is even, $n' = n-1$, because $n-1$ is odd and there is basic shifted declustering layout for $(k, n-1)$ configuration. If $k > 3$, n is non-prime, then n' is the largest prime number smaller than n.

Now the value of $P^{(l)}$ is

$$P^{(l)} = \frac{S_{sd}(n',k,l)}{C(n,l)} \tag{4.13}$$

Similar to the basic solution, if $l = k$, $P^{(k)} = 1 - n(n'-1)/2C(n,k)$. Because if first failed disk is chosen, there are $(n'-1)/2$ types of distances to decide the other $k-1$ failed disks to cause data loss.

When $l = k$, the value of $P^{(k)}$ can be explicitly given:

$$P^{(k)} = 1 - n(n' - 1)/2C(n, k) \tag{4.14}$$

where $n' = n$ for the basic solution; $n - 1$, for the extended solution if $k = 3$; n' is the maximum prime number less than n for the extended solution if $k > 3$.

In shifted declustering layout, the equations for $P^{(l)}$ are difficult to summarize. This is because the combination behavior is evaluated by "fixed distances", which varies from 1 to $\lfloor n'/2 \rfloor$, so it is not easy to distinguish independent subsets for the purpose of generalizing the behavior. For this reason, we have difficulty in abstracting the expression of $P^{(l)}$ explicitly.

copyset Replication Layout

Copyset replication is a novel technique that provides a near optimal solution between scatter width and the number of copsets [46]. Figure 4.3 is an example of copyset replication technique with a scatter width equals 4 and the number of replicas for each chunk is 3. In this example, there are 12 chunks in each disk and 3 replicas for each chunck. Take chunk (0,0)in disk 0 as an example, the other two replicas (0,1) and (0,2) are either in disk 1 or in disk 2. Figure 4.3 illustrates that the chunks of disk 0 is either replicated on disk 1 and disk 2 or replicated on disk 3 and disk 6. In this case disk {0,1,2} and disk {0,3,6} are formed as copysets. Based on this regulation, the total combination of copysets of these nine disks are {0,1,2}, {0,3,6}, {1,4,7}, {3,4,5}, {6,7,8} and {2,5,8}. Accordingly, the system will only lose data if and only if it loses a whole copyset. However, copyset replication method trades off the probability of data loss with the amount of lost data in each incident.

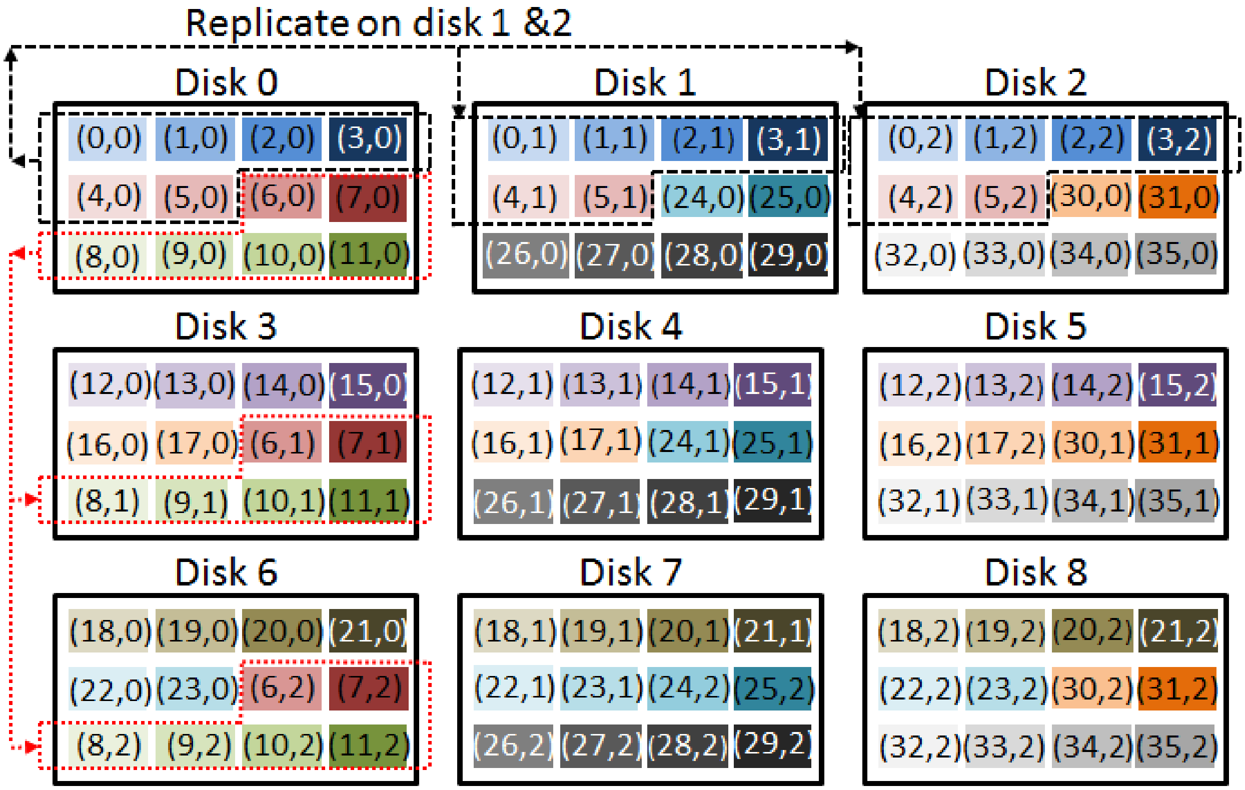


Figure 4.3: Copyset Replication layout

In other word, the storage system may not lose data frequently as comparing with shifted or random declustering layout but may lose a larger amount of data if node failure happens.

Therefore, if three nodes fail at the same time, the probability of data loss in this specific example is:

$$\frac{numberofcopysets}{totalnumberofcopysets} = \frac{6}{84} = 0.07 \tag{4.15}$$

Explicitly, if k-way replication has been used in this copyset technique, each copyset group must contain at least k nodes to ensure that all replicas of the primary chunks will be included. As we mentioned before, the data loss will only happen when a whole copyset has been lost. Thus, if l disks fail and l is an integer multiples of k,the probability of no-data loss after the l-th disk failure is:

$$P^{(l)} = 1 - \frac{2^{\frac{n}{k}}}{C(n,k)} \frac{l}{k} \tag{4.16}$$

Otherwise, the probability of no-data loss after l disk fails will be:

$$P^{(l)} = 1 - \frac{2^{\frac{n}{k}}}{C(n,k)} \lfloor \frac{l}{k} \rfloor \tag{4.17}$$

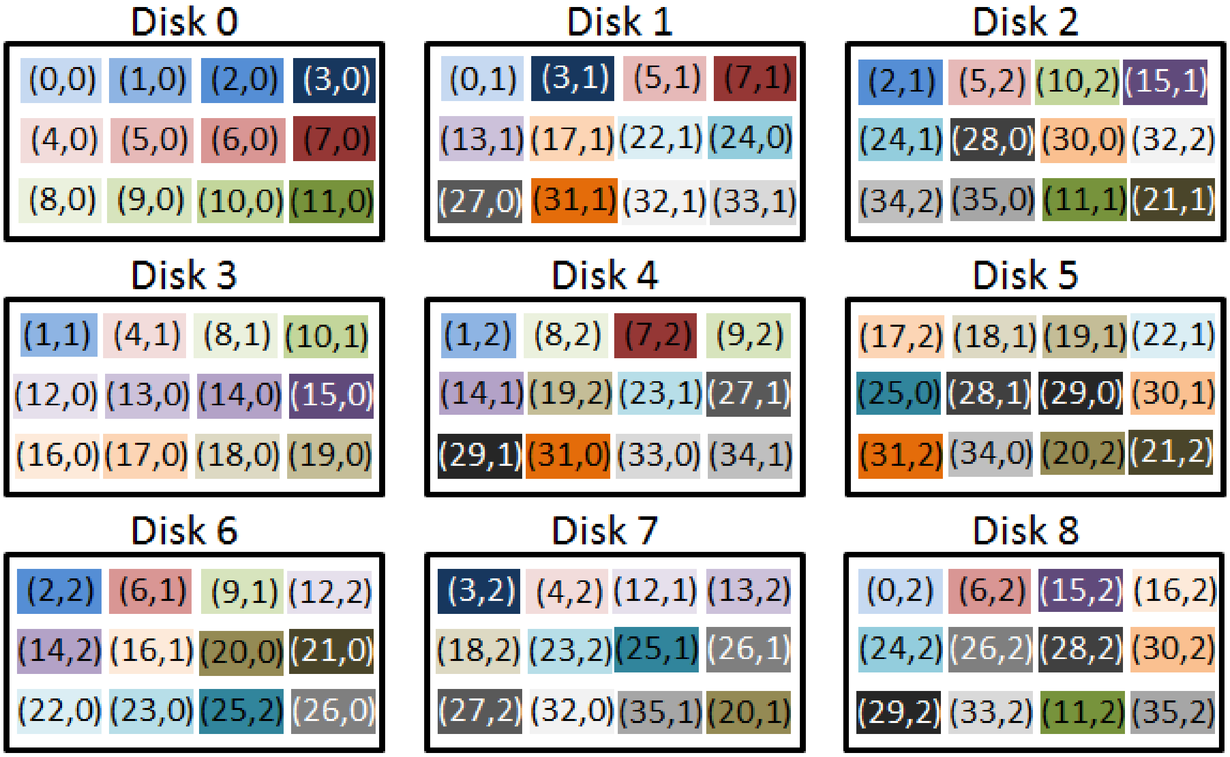


Figure 4.4: Random Declustering layout

Random Declustering Layout

Random declustering layout distributes data blocks according to given randomization algorithms, which map the key (or the index) of a data block to a position in the storage system. Assume that each data block has a unique key, and the blocks within a redundancy group have different keys. Ideally, a good randomization algorithm distributes data in a balanced manner. If a disk fails, all other disks should share the same amount of traffic redirected from the failed disk. Figure 4.4 illustrates an example of random declustering layout.

The probability of losing (or not losing) data upon l failed disks depends not only on l, the number of disks (n), and the number of replicas in each redundancy group (k), but also on the number of redundancy groups (r). The number of redundancy groups is a function of n, k, the used capacity of disks (S), and the size of a data unit (s). We assume that the used space of each disk is the same, and the size of a data unit is fixed. The number of redundancy groups can be derived as $r = \dfrac{S \times n}{s \times k}$, where $S \times n$ means the total used space for data storage, and $s \times k$ is the space used by one redundancy group.

There are $C(n,k)$ ways to distribute a redundancy group. We assume that an ideal random declustering algorithm has the ability to distribute redundancy groups to at most combinations of disks as possible. With this assumption, if $r < C(n,k)$, the redundancy groups are distributed to r combinations of k disks. If $r \geq C(n,k)$, all $C(n,k)$ combinations are used for distributing redundancy groups. Upon l disk failure, the probability of losing data is the probability of that the k failed disks is one combination of the r combinations used for distributing redundancy groups. When $l = k$, the probability of not losing data is:

$$P^{(k)} = \max(1 - r/C(n,k), 0) \tag{4.18}$$

For $l > k$, due to the lack of mathematical regulations to describe the behavior of random declustering, we use sampling techniques to obtain the values of $P^{(l)}$.

Aggrestive parallel recovery and the repair rate μ_l

Recovery is commonly performed in two different ways: off-line and on-line [64]. In large-scale storage systems, on-line recovery is more acceptable, because the system will not pause service while the recovery process is undergoing. In this work, we assume that on-line recovery is used and each disk will dedicate a certain amount of bandwidth for recovery if the disk is involved in a recovery process. This assumption is reasonable, because such bandwidth allocation schemes are widely used in storage quality of service (QoS), and can cooperate with other mechanisms like latency control, burst handling, etc. to guarantee a minimum compromise of storage performance[63]. If recovery is considered as an event that requires a priority in a storage system, similar policies can be used to provide a certain amount of sources for this process.

For simplicity, we assume a fixed bandwidth per disk in recovery (*defined as recovery bandwidth per disk*), optimizing the storage QoS is out of our research emphasis of this work. Since in current disk drive architectures there is only one read/write channel, at any time a surviving disk is either under normal service or under recovery. The concurrency of normal service and recovery at a course grained time scale can be obtained in a time sharing manner at a fine grained time scale. For example, we are able to get a 10 KB/sec recovery bandwidth usage by controlling a disk to transfer 10 KB recovery data per second, and to serve normal requests in the remaining time.

Let $\mu = b_r/S$, where S is the capacity of a disk used for storage, and b_r is the recovery bandwidth per disk, so μ is the rate of sequentially recovering one failed disk. We assume an aggressive recovery scheme: for a failed disk, its data replicas can be found from multiple other disks, and as many as possible disks sharing redundant data with the failed disk will be involved in the recovery process. When a disk fails, a new disk is added, but the aggressive recovery does not recover the failed disk's data to the new disk sequentially. Instead, the system will take advantage of spare space on all surviving disks and the new disk to recover the data as fast as possible. First the data is reconstructed and written to the available spare space, then the recovered data is moved to the new disk in the background to restore the system to the original layout before failure. As long as the data of the failed disks is restored somewhere in the system, the failed disks are considered as recovered.

With the above assumption, the recovery rate upon l-disk failures (μ_l) is proportional to the number of disks which can provide data for recovery. We define the disks providing data for recovery as *source* disks, and the disks which the recovered data is written to as *target* disks. At any moment, a disk can only be a source disk or a target disk. If the number of source disks surpasses half of the total number of disks, we consider $\lfloor n/2 \rfloor$ as the number of source disks, because at most $\lfloor n/2 \rfloor$ source-target disk pairs can be formed. Upon l disk failures, we assume l blank disks are added, when $l < n/2$, the possible maximum source disks is still$\lfloor n/2 \rfloor$; when $l \geq \lceil n/2 \rceil$, the possible maximum source disks is $n - l$, which is the number of all surviving disks. Accordingly, we can get an upper bound of recovery rate: $\min(\lfloor n/2 \rfloor \mu, (n-l)\mu)$.

However, the upper bound recovery rate may not be achievable in non-ideal layouts, because the replicas of data on the failed disk may be limited to a small number of surviving disks. As a result, the transfer rate can be expressed as the product of a factor x and the value μ, where x is the smaller one among $\lfloor n/2 \rfloor$, $n - l$ and the number of source disks. In addition, with different failed disk combinations, the number of source disks differs. With this assumption, we can conclude the ranges of μ_l for each layout:

In shifted declustering and random declustering layouts, all surviving disks can provide data for recovery, so the recovery rate is always $\min(\lfloor n/2 \rfloor \mu, (n-l)\mu)$.

For random declustering, we make the assumption that all disks across all nodes have an equal probability of being selected to become a replica holder. This is a slight deviation from the Google File System [62] (and Hadoop's HDFS [48]) which incorporates rack position awareness and thus limits the nodes and drives that could potentially be selected by the random placement algorithm. This assumption simplifies the analysis of the system while still providing a model that could be used within the locality group specified in these real world systems.

For copyset replication,$\mu_l \leq \min((k-1)l\mu, \lfloor n/2 \rfloor \mu, (n-l)\mu)$, and n must be a multiple of k.

The highest recovery rate is achieved if the l failed disks are in as many redundancy disk groups as possible. This includes the following situations:

- If $l \leq n/k$, the highest recovery rate is achieved if the l failed disks are in l different redundancy disk groups, because for each failed disk, $k-1$ surviving disks within its redundancy disk group are its source disks. In this case, the recovery rate is $(k-1)l\mu$. In addition, if $(k-1)l\mu > \lfloor (n-l)/2 \rfloor \mu$, the highest recovery rate is $\lfloor (n-l)/2 \rfloor \mu$, since it is the upper bound as analyzed above.

- If $n/k < l \leq n(k-1)/k$, the highest recovery rate is achieved if the l failed disks are in all n/k redundancy disk groups, so all surviving disks can be source disks. Therefore, in this case, the recovery rate reaches the upper bound, which is $\min(\lfloor n/2 \rfloor \mu, (n-l)\mu)$.

As a result, the highest recovery rate is $\min((k-1)l\mu,$
$\lfloor n/2 \rfloor \mu, (n-l)\mu)$, so $\mu_l \leq \min((k-1)l\mu, \lfloor n/2 \rfloor \mu, (n-l)\mu)$.

In contrast, the lowest recovery rate is achieved if l failed disks are in as few redundancy disk groups as possible. This happens when the l failed disks are limited within $\lceil l/(k-1) \rceil$ redundancy disk groups. In each of $\lfloor l/(k-1) \rfloor$ redundancy disk groups, $k-1$ disks fail, and the failed disks can be recovered by only one surviving disk. In the remaining redundancy disk group (if $\lceil l/(k-1) \rceil \neq \lfloor l/(k-1) \rfloor$), $l \bmod (k-1)$ disks fail, and they can be recovered by $k-(l \bmod (k-1))$ disks.

Comparison between Copyset,Shifted and Random Declustering

Some current enterprise-scale storage systems adopt random data layout schemes to distribute data units among storage nodes [62, 48]. There is also theoretical research on algorithms for random data distribution in large-scale storage systems [66]. Random declustering attracts people's interests, because it brings a statistically balanced distribution of data, thus providing optimal parallelism and load balancing in both normal and degraded mode[1].

Compared to random declustering, Shifted declustering layout deterministically guarantees the optimal parallelism and load balancing for replication based storage systems. For a large data set, both random and Shifted Declustering deliver comparably high performance due to optimal parallelism, but they may provide different reliability. In Shifted Declustering layout, $P^{(k)}$ (probability of not losing data when the system has k failed disks) is independent from the number of redundancy groups. It is given as Equation (4.14) in Section 4. We also find that $P^{(k)}$ in random declustering layout is closely related to the number of redundancy groups (r) in the system, and give it in Equation (4.18) in Section 4.

Copyset replication is able to achieve the best probability of no data loss comparing with random and Shifted Declustering layout due to its carefully replication of the data blocks which largely reduce the number of redundancy groups, so that minimize the probability of data loss.

[1] If there are disk failures in the system, but the data set is still complete, the service work load which is supposed to be processed by the failed disks is directed to surviving disks. This system status is called degraded mode.

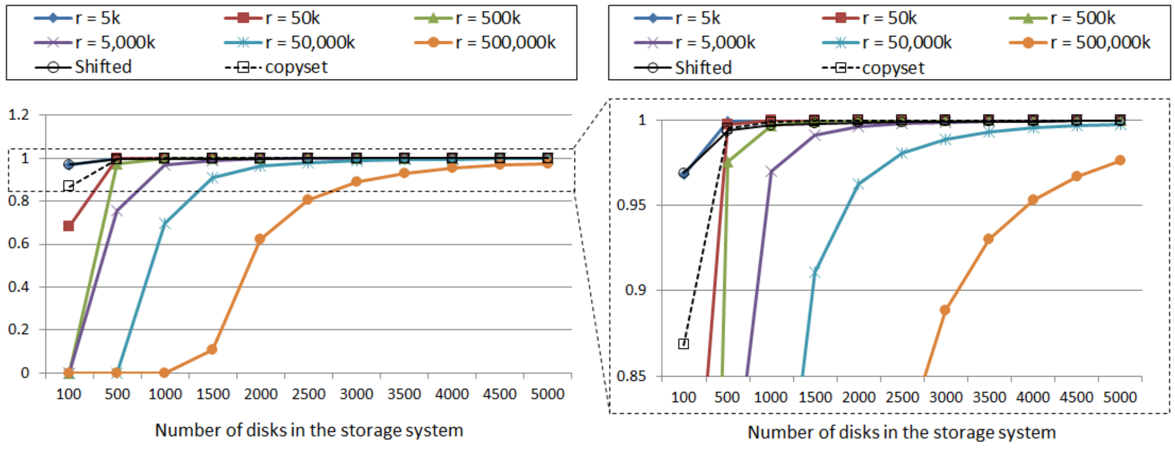


Figure 4.5: The comparison of $P^{(k)}$ between copyset,shifted and random declustering

From Equations (4.14) and (4.18), we can see that as long as n and k are determined, $P^{(k)}$ of Shifted Declustering is fixed. While $P^{(k)}$ of random declustering is negatively linear with r. This indicates that the more redundancy groups there are in random declustered storage systems, the higher the probability of data loss upon k disk failures. Additionally, in Section 4, we show that these two layouts have the same potential for parallel recovery, so the reliability can be directly reflected by the value of $P^{(k)}$.

In Figure 4.6, we quantitatively compare the change of $P^{(l)}$ with the number of disks (n) and the number of redundancy groups (r). We assume $k = 3$, the number of disks varies from 100 to 5,000, and the number of redundancy groups varies from 5,000 to 5×10^9. The left diagram demonstrates the dependence of $P^{(k)}$ on n and r in random declustering. Each line reflects the value change of $P^{(k)}$ for a certain r and a varying n. We can see that with a fixed number of disks, the more redundancy groups the system has, the smaller is $P^{(k)}$, thus the easier the system lose data upon k disk failures. We enlarge the portion near $P^{(k)} = 1$ to the right diagram, and add the $P^{(k)}$ values of both shifted declustering and copyset. When the number of disks in the storage system is larger than 100, we can see that $P^{(k)}$ of shifted declustering and copyset are constantly approaching 1 with the increase in the number of disks, so the larger the system scale, the lower the probability that the system will lose data upon k disk failures. However, when the storage system only has 99 disks, the $P^{(k)}$ of copyset is worse than shifted declustering because it is sensitive to the number of disks as well as the scatter width.

There is a break-even point of r (denoted by r^*) that random and shifted declustering have the same value of $P^{(k)}$. It can be found by making Equations (4.14) and (4.18) equal to each other, and solving the following equation:

$$1 - r^*/C(n,k) = 1 - n(n' - 1)/2C(n,k) \tag{4.19}$$

Equation 4.19 yields $r^* = n(n' - 1)/2$. For a given number of disks (n), if $r > r^*$, shifted declustering has higher reliability, and vise versa.

Considering a storage system with 1,000 disks, configured as in the Google File System: each data chunk has three copies, and each copy is 64 MB. The break-even number of the redundancy groups $r^* = 499,000$, this only allow the system to store at most about 96 TB (96 GB per disk) data when using the random declustering layout, otherwise the system is less reliable than with shifted declustering layout. Notice that 96 GB per disk is a small amount of capacity in today's hard drives. For larger storage system with more disks, the data size per disk will be even smaller for the random declustering layout to provide the same reliability as achieved with the shifted declustering.

Simulation Results

Methodology

We use the SHARPE package [71] for the simulation of system reliability. The simulation time is 10 years, in steps of a month. We simulate a storage system of 999 disks, configured as three-way replication, with shifted ,random and Copyset layouts.

For the disk failure rate λ, we use 3% ARR (annual replace rate), which corresponds to $\lambda = 0.25\%$ per month. This value is from Schroeder et al.'s observation [72]: in real-world large storage and computing systems, the ARR of hard drives is between 0.5% and 13.8%, and 3% on average. It is much higher than the failure rate provided by the most vendors, a MTTF (mean time to failure) of a single disk from 1,000,000 to 1,500,000 hours, which corresponds to an AFR (annual failure rate) between 0.58% and 0.88% [72].

The size of a data replica was set to 64MB, which is a typical data block size for Hadoop File system(HDFS). The capacity of a disk used for the storage is 1TB, which is a normal size of a hard disk drive used for storage systems. In our simulation, all the disks are considered to be hard disk drives (HDD). Other storage devices such as SSD are out of the research scope of our work.

The recovery rate upon l failed disks is determined by the product of the available source disks and the sequential recovery rate offered per disk (μ). We will simulate the reliability for 10 years with different values of μ. In the Section 4, we give a range of the recovery rate upon l failed disks (μ_l) for standard mirroring and chained declustering, but we only take their highest recovery rate in the simulation.

The values of parameters in the simulation are listed in Table 4.1.

Table 4.1: Parameters in Simulation

Parameter and description	Value
n: Number of disks in the system	999
k: Number of ways of replication	3
l: Number of failed disks	3 to 666
$P^{(l)}$: Probability of no data loss upon l-disk failure	See Section 4
λ: Disk failure rate	0.25% per month
λ_l: Transition rate from l-disk failure to $l+1$-disk failure **without** data loss	$(n-l)\lambda P^{(l+1)}$
γ_l: Transition rate from l-disk failure to $l+1$-disk failure **with** replica lost	$(n-1)\lambda(1-p_b)$
δ_l: Transition rate from l-disk failure to $l+1$-disk failure **with** data loss	$(n-l)\lambda(1-P^{(l+1)})$
b_r: Reserved bandwidth for recovery per disk	0 KB to 10 MB
S: Capacity of a disk used for storage	1 TB
μ: Sequential recovering rate	b_r/S
μ_l: Recovery rate from $l+1$-disk failure to l-disk failure	See Section 4
s: Size of a data replica	64 MB
r: Number of redundancy groups	5,455,872
N: Number of the total data blocks per disk	16,384

Sampling Procedures

The sampling procedures for a storage with 999 disks and three-way replications with shifted and random declustering are as follows:

- For shifted declustering, the values are obtained by the following simulation procedure: for any l between 3 and 666,[2] randomly generate 10,000 vectors of l numbers between 0 and 998 as failed disks; for each vector, check whether this particular failed l disks causes data loss according to the criteria discussed in Section 4. $P^{(l)}$ is estimated by the result of dividing the number of vectors that do not cause data loss by 10,000. This process is repeated 5 times, and the average $P^{(l)}$ is used as the final value.

[2] For 999-disk three-way replication storage, if the number of failed disks is more than 2/3 of the total number of disks, data loss becomes a definite event.

- For copyset replication, we assume the capacity for storage is 1 TB per disk, the data replica size is 64 MB, so the 999-disk system can store 5,455,872 redundancy groups in total. We generate 5,455,872 random vectors with three numbers between 0 and 998 to represent where the three replicas in each redundancy group are distributed.

- For random declustering, we assume the capacity for storage is 1 TB per disk, the data replica size is 64 MB, so the 999-disk system can store 5,455,872 redundancy groups in total. We generate 5,455,872 random vectors with three numbers between 0 and 998 to represent where the three replicas in each redundancy group are distributed. The following procedure is similar to that used for shifted declustering, for any l between 3 and 666, randomly generate 10,000 vectors of l numbers between 0 and 998 as failed disks; for each vector, check whether this particular failed l disks cause data loss by checking whether it contains three elements of any vector that is used to store a redundancy group. $P^{(l)}$ is estimated by the result of dividing the number of combinations that do not cause data loss by 10,000. Repeat the process 5 times, and compute the average $P^{(l)}$.

Probility of No-data-Loss

With the methodology introduced in Section 4, we calculate the value of $P^{(l)}$ (the probability of no data loss in case of l disk failure) in a 999-disk system ($n = 999$) with 3-way replication ($k = 3$), configured with shifted and random declustering layout schemes by Matlab [69]. The value of $P^{(l)}$ is illustrated in Figure 4.6.

Theoretically, a 999-disk 3-way replication system can tolerate as many as 666 failed disks, but $P^{(l)}$ drops close to 0 much earlier than l approaching 666, as shown in Figure 4.6.For shifted declustering and random declustering layouts, $P^{(l)}$ drops to 0 at 28 and 13, respectively. For the same number of failed disks, We can see that our shifted declustered layout is always has the higher possibility of not losing data comparing with random declustering.

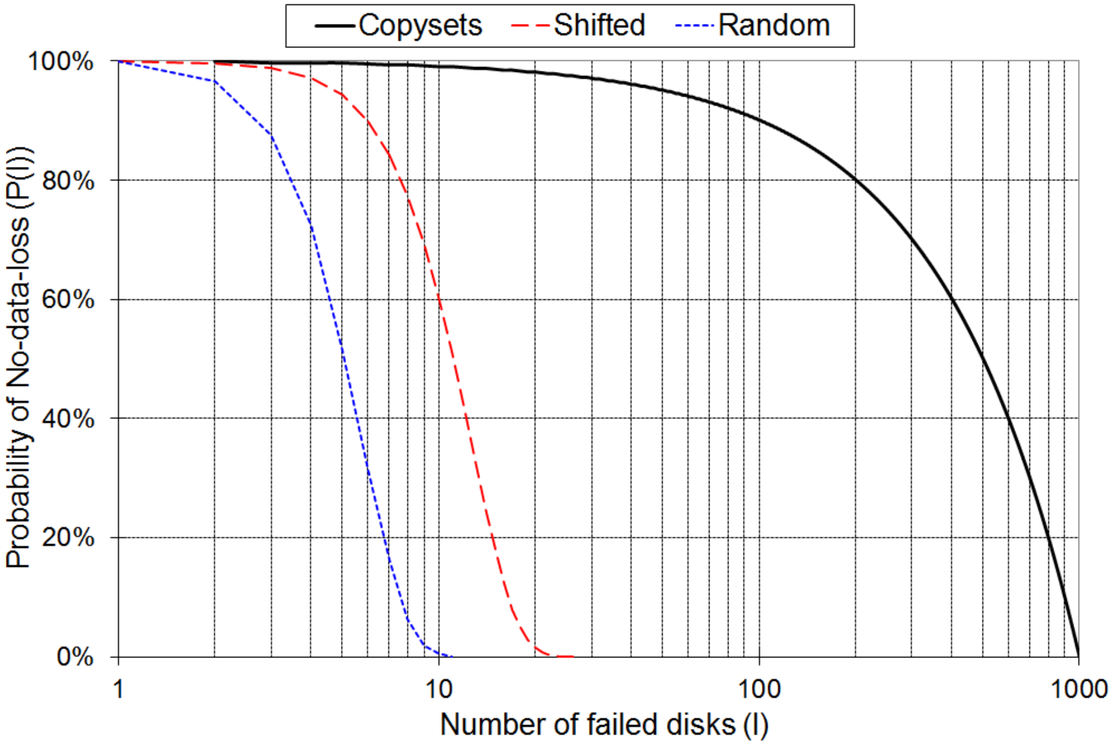


Figure 4.6: Probability of no data loss in a 999-disk system ($P^{(l)}$)

System Reliability without Recovery

If the system does not apply any recovery mechanism, the system reliability can be obtained by assigning $b_r = 0$. The system reliability of shifted declustering and random declustering layouts are illustrated in Figure 4.7. The physical meaning of system reliability at time t (represented by $R_{\text{system}}(t)$) is the probability of the system surviving until the time point t. The reliability without recovery is a direct reflection of $P^{(l)}$. Thus, a higher probability of the system not losing data results in a higher overall system reliability rating due to it being more difficult to enter the failed state. We can see that after the third month, the reliability is almost 84% and 60% for shifted declustering and random declustering respectively. This reliability results has the same tendency of the $P^{(l)}$ of these two layouts. Without considering the recovery, the Copyset layout achieve the best reliability result because of its high probability of no-data-loss.

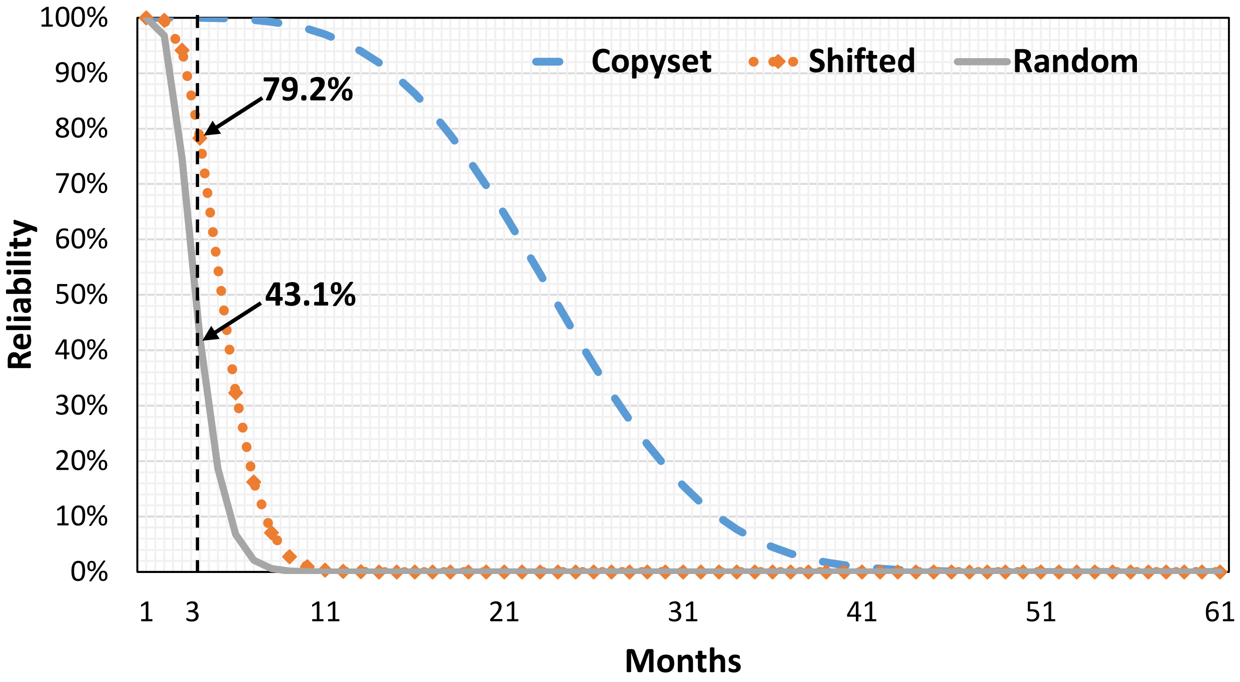


Figure 4.7: System reliability of a 999-disk system without recovery

With this specific configuration, shifted declustering has a significant advantage in terms of reliability to random declustering. This result is consistent with our analysis in Section 4.

System Reliability with Recovery

The previous section discusses the storage system reliability without data recovery in place. It reflects how reliable the system will be with different data layouts, when disk failures are not detected, or replicas on failed disks are never recovered. In this section, we simulate the system reliability of different layouts with the aggressive recovery schemes introduced in Section 4.

With the aggressive recovery schemes, shifted declustering layout has the best reliability for a given recovery bandwidth per disk for recovery (b_r) among all layouts. The simulation results in Section 4 show that with recovery, shifted declustering layout's system reliability is better than the random declustering and the Copyset. This is reflected in the fact that the shifted declustering layout is slower to transmit from non-data-loss states to the data-loss state than the other two. Additionally, although the Copyset layout has a higher probability of no-data-loss, the amount of data lost is huge compared with the other two layouts. Therefore, when we take data recovery into consideration, the system reliability will decrease. As a result, even these three layouts have the same potential of optimal parallel recovery, the reliability with recovery of shifted declustering exceeds it of the other two.

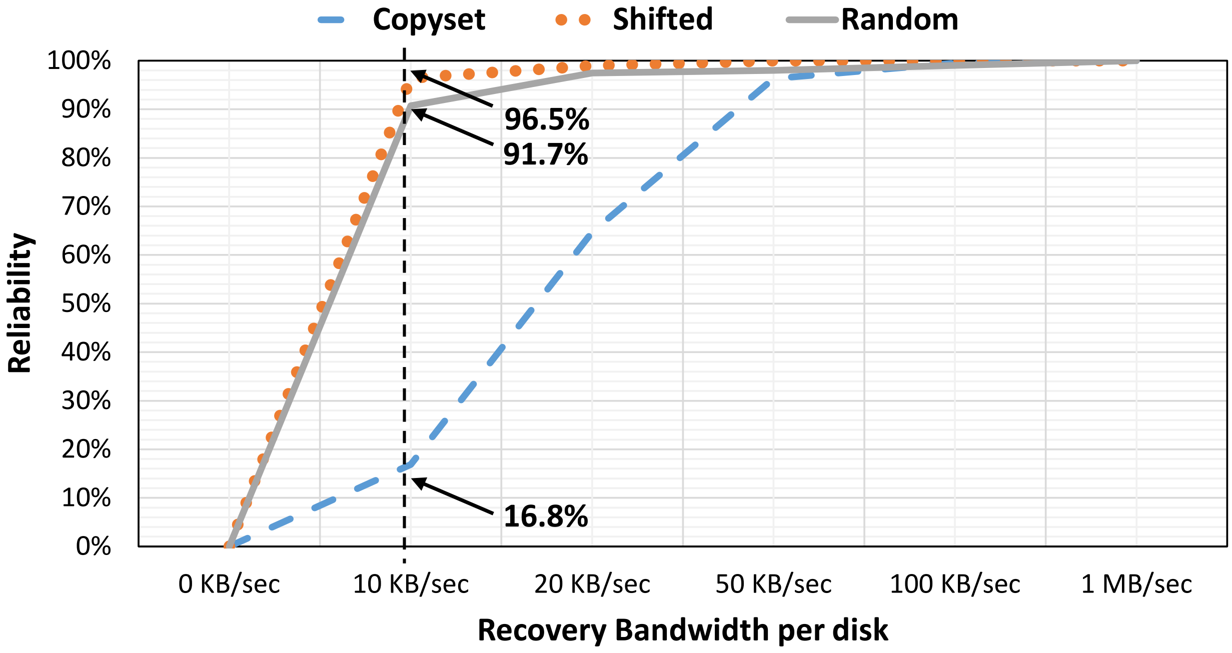


Figure 4.8: Reliability after 5 years

We can see that from Figure 4.8, shifted declustering with 10 KB/sec recovery bandwidth per disk (5 MB/sec accumulative recovery bandwidth for 999-disk system) obtains more than 99% reliability after 5 years, while the Copyset layout requires more than 100 KB/sec recovery bandwidth per disk to achieve the same reliability.

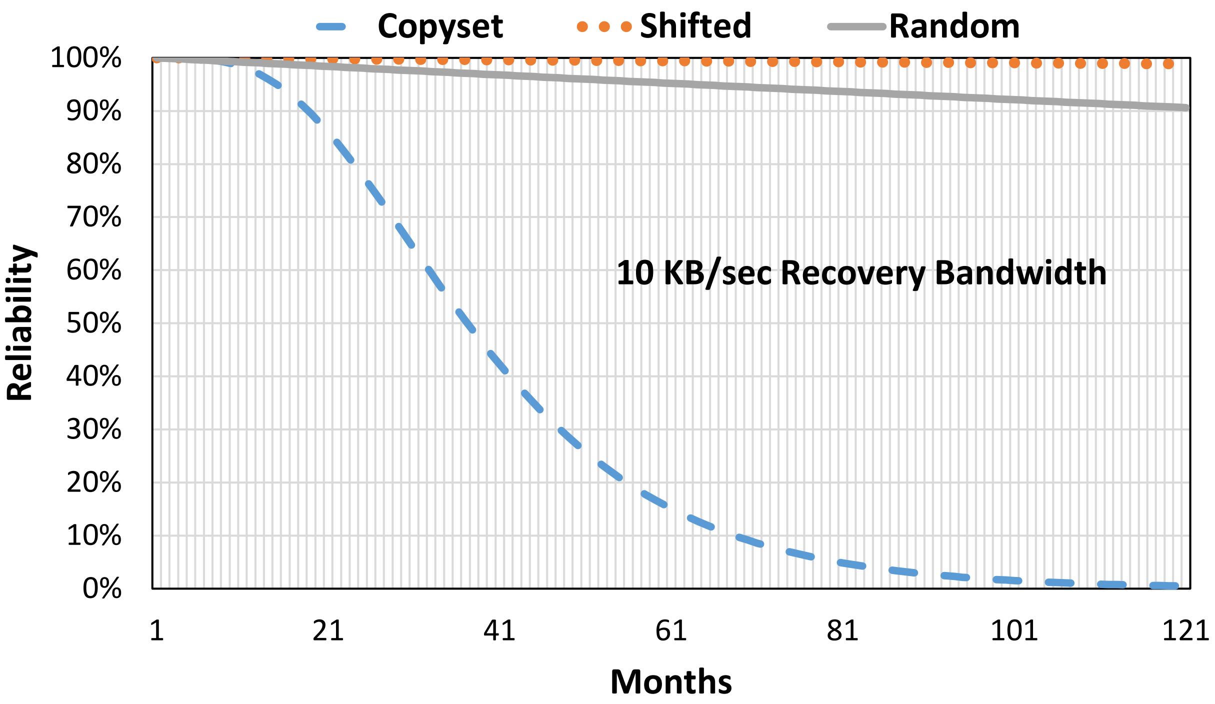


Figure 4.9: System reliability with 10 KB recovery bandwidth per disk

Similarly, to reach the same reliability, the random declustering layout has to use 20 KB/sec. It is true that the reliability can be improved by increasing the recovery bandwidth to obtain a higher recovery rate. Nonetheless, as long as the disk transfer bandwidth is fixed, higher recovery bandwidth means lower bandwidth for normal services, so higher recovery bandwidth will drag down the normal service performance of the disks involved in recovery. Obviously, for the same reliability goal, the burden on surviving disks of shifted declustering is the lightest, because the service load is balanced among all surviving disks, and the bandwidth used by each disk for recovery is low. In comparison, for the Copyset, the service load on the source disks is heavier than the other surviving disks, because they need to serve requests which are supposed to be served by the failed disks besides normal service; the recovery bandwidth per disk is also higher.

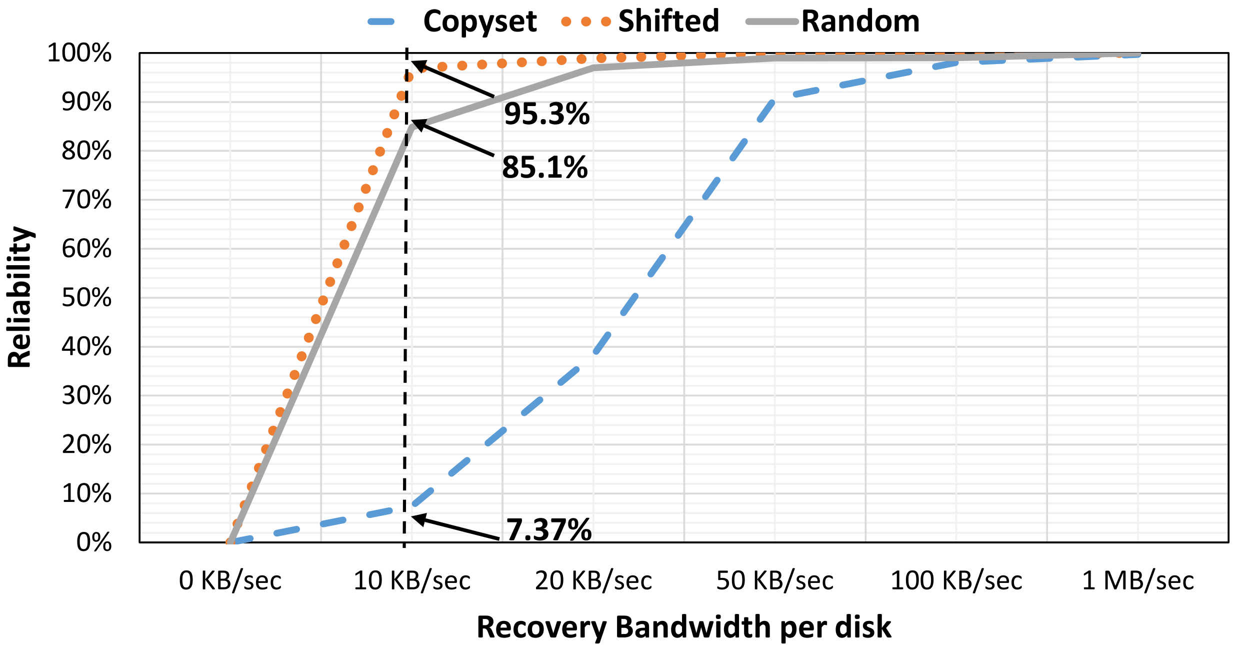


Figure 4.10: Reliability after 10 years

These two reasons result in a load imbalance more severe in Copyset. If we want to achieve the same recovery bandwidth of 120 MB/sec, which is the highest recovery bandwidth obtained in the Panasas parallel file system [76] with around 120 disks, corresponding to 999 MB/sec for a 999-disk system three-way replication, for shifted declustering, about 2 MB/sec recovery bandwidth per disk is enough; while for Copyset, about 500 MB/sec recovery bandwidth is needed, which is almost impossible to obtain with current storage technology.

Figure 4.9 shows the system reliability during the first ten years, with the recovery bandwidth per disk set to 10 KB/sec ($b_r = 10$ KB/sec). Currently the sustained transfer rate of hard drives can be higher than 100 MB/sec[3], so 10 KB/sec dedicated to recovery has little impact on the regular

[3] The Barracuda 7200.11 Serial ATA (released May 2008 by Seagate) is specced to have a sustained transfer rate of 105 MB/sec [52].

service workloads. We can see that with such a small recovery bandwidth per disk, the shifted declustering layout obtains a near 99% reliability even after ten years (Figure 4.10).

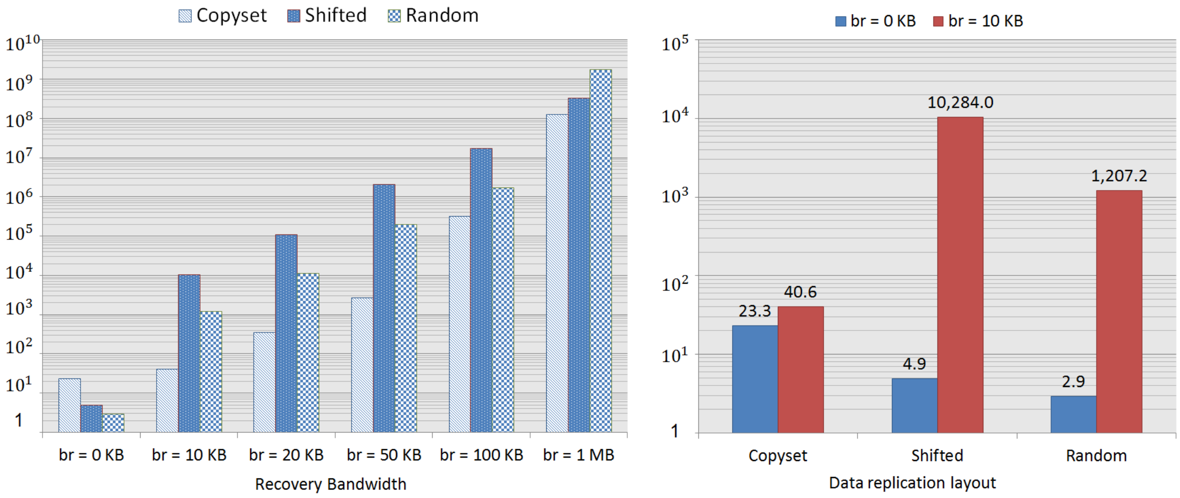


Figure 4.11: Mean Time to Data Loss

With the same recovery bandwidth, the reliability of other layouts is lower. To reach the same reliability as shifted declustering, other layouts need more recovery bandwidth per disk.

System reliability with and without considering probability of replica lost

In order to show the accuracy of our proposed reliability model, we conduct a comparison between the system reliability with and without considering the probability of replica lost. We run the simulation to obtain the 5 years system reliability results for all the three replication methods by setting the system recovery bandwidth to 10KB/sec. In the first round of simulation, we do not considering about the probability of replica lost with setting the initial system reliability equals to 1. In the second round of simulation, we taken probability of replica lost into account by utilizing our proposed model and obtain a more precise outcomes.

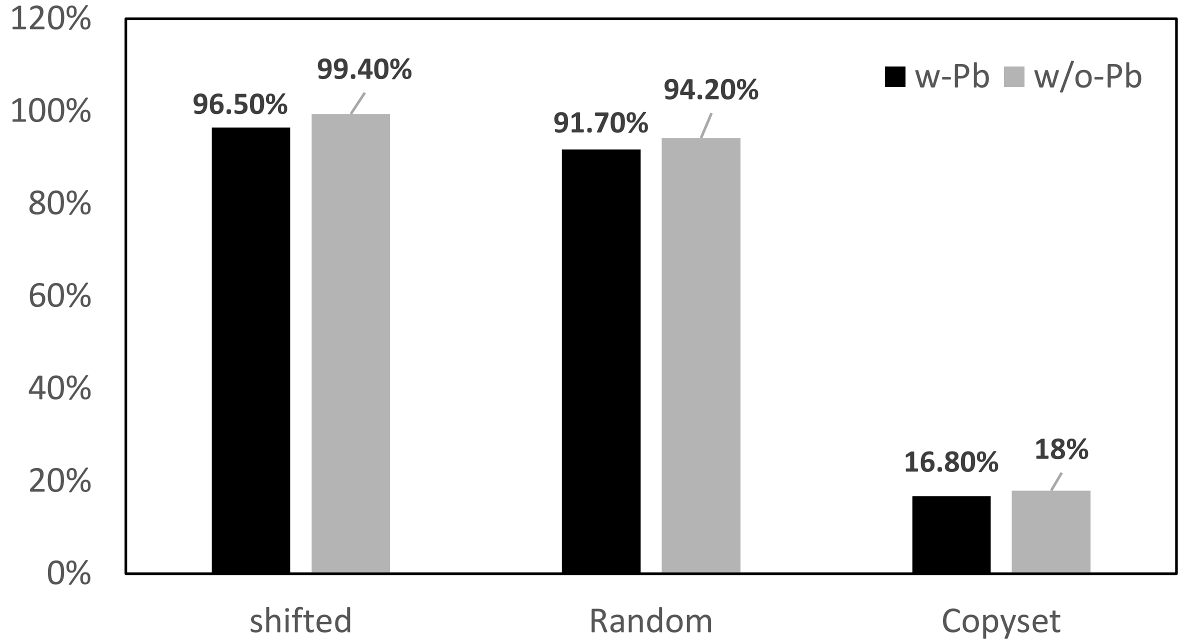


Figure 4.12: Average system Reliability with and without considering probability of replica lost, recovery bandwidth = 10KB/sec

The results have been illustrated in Figure 4.12. Based on the results, we can see that the probability of replica lost do have an influence on the overall system reliability. For shifted, random and copyset replication methods, the system reliability result with considering the probability of replica lost are 96.5%, 91.7% and 16.8% respectively. However, we obtain 99.4%, 94.2% and 18% for the above three replication methods by introducing the probability of replica lost. The differences among these two set of experimental results are 2.9%, 2.5% and 1.2% respectively. The reason that the overall system reliability is low for copyset replication approach because of the selected low recovery bandwidth (10KB/sec), which was discussed in Section 4.

Mean Time to Data Loss

Mean time to data loss (MTTDL) is the expected duration between the state of all disks functioning in a storage system (state 0 in Figure 4.1) and the state of data loss (state F in Figure 4.1). Figure 4.11 demonstrates the simulation results of mean time to data loss based on the parameters listed in Table 4.1.

The left bar in each group is the MTTDL without recovery. The Copyset lasts the longest, 23.3 months. We can see that without recovery, even the most "reliable" system is unacceptable, because data in that system will start to lose approximately two years. The right bar in each group is the MTTDL with 10 KB/sec recovery bandwidth per disk. With the advantage of parallel recovery, shifted declustering and random layouts have higher accumulative recovery bandwidth than the Copyset layout. As a result, the MTTDL is improved by 253 times and 29 times in the shifted declustering and random layout.

In addition, the difference in reliability of shifted declustering and random declustering layouts are not very significant. For example, after 5 years, shifted declustering layout has a 99% reliability, and random declustering layout has a 95% reliability; after 10 years, the values becomes 98% and 90%. However, the difference of system reliability yields significant improvement in terms of MTTDL in the shifted declustering layout compared to the other two layouts. In the shifted declustering layout, an almost 8.5 times higher MTTDL than the second ranked layout, random declustering layout, is observed in our simulation environment; it is also 254 times higher than the Copyset. These result indicates that the shifted declustering layout achieves the highest reliability given a fixed resource budget among all layouts in comparison. On the other hand, it has the maximum flexibility to be tuned to a desired reliability goal.

Conclusions

In this work, we have modeled the reliability of multi-way replication storage systems with different data layout schemes. We make the following conclusions:

- Modeling of the system reliability should not only taking data loss probability and recovery bandwidth into account, but also need to considering about the probability of replica lost in order to obtain an accurate result.
- The reliability of random declustering layout is highly dependent on the number of redundancy groups in the system. With the increase of redundancy groups and/or the number of disks in the system, the reliability of random declustering drops.
- The shifted declustering is less sensitive to the scale of the storage system comparing with random declustering. With the same resource provided for recovery per disk, the shifted declustering layout achieves almost 100% reliable that lasting for 10-years long period. In particular, the data integrity of the shifted declustering layout lasts 85% times longer in our simulation than random declustering layout.
- The Copyset replication method obtained the best system reliability due to its carefully arrangement of the data blocks. However, with considering of the recovery bandwidth, the system reliability has been greatly affected especially when the bandwidth is low.
- Our study on both 5-year and 10-year system reliability equipped with various recovery bandwidth settings shows that, the shifted declustering layout surpasses the two baseline approaches in both cases by consuming up to 83 % and 97% less recovery bandwidth for copyset, as well as 5.2% and 11% less recovery bandwidth for random layout.

As random declustering layout is widely adopted by enterprise large-scale systems configured with multi-way replication, shifted declustering layout is a promising alternative for its proved optimal performance and high reliability with low recovery overhead.

CHAPTER 5: AN ENERGY-EFFICIENT JOB SCHEDULER FOR SUPERCOMPUTERS

Scheduler Design

In this section, we introduce the design, analysis, and optimization of our power-aware job scheduler. In the first part, we present our job scheduling polices. In the second part, we introduce a mathematical model of the incoming jobs so as to have a theoretical understanding of the overall problem. In the last part, we show the controller design of our Scheduler.

Job scheduling polices

The job scheduler needs to make decisions when scheduling the incoming jobs. To decide the running order and allocation of corresponding resources, we have to collect the following information for each job.

- The job arrival time;
- The job start time;
- The job finish time;
- The user estimated run-time;
- The job size
- The requested number of nodes;
- The requested number of processors;

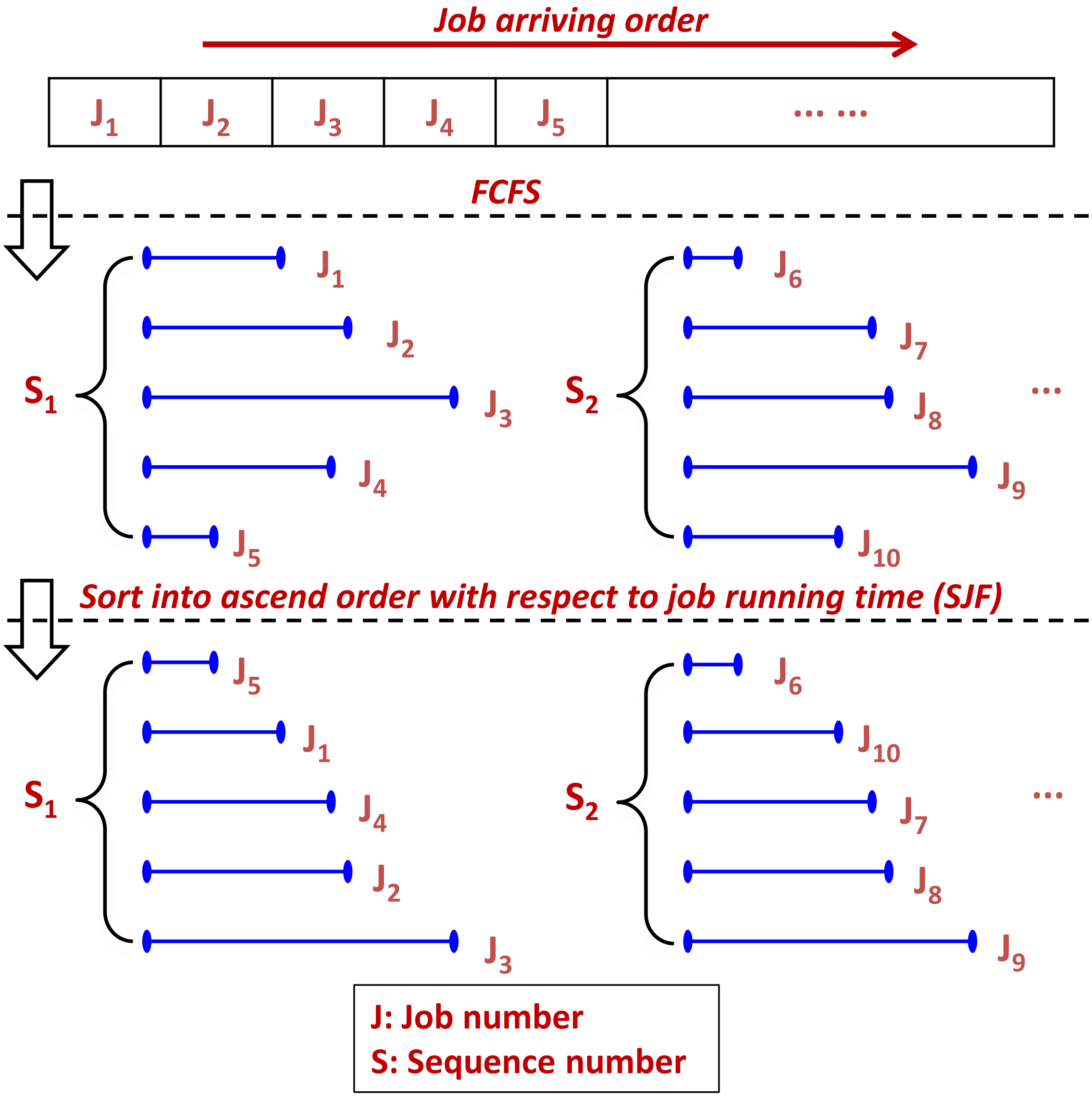


Figure 5.1: The illustration of our proposed scheduling algorithm

In order to maintain user fairness and achieve a better mean response time, we take advantages of both the FIFO (First In First Out) and SJF (Shortest Job First) scheduling policies to generate a new scheduling algorithm to balance the system utilization as well as the quality of service. The visual explanation of our proposed scheduling algorithm is presented in figure 5.1. Specifically, instead of considering the incoming jobs one-by-one, we consider a small sequence of jobs arriving earlier and then make the decision as to which job should be processed first. When we look into the small sequence (the length of the sequence will be discussed in the section 5), the jobs are sorted into ascending order of their run-times. This means that the job with the shortest run-time in the sequence will be scheduled to first and so on. This shortest time first policy is implemented to ensure a better system response time. The priority scheduling policy is also considered if there exists any jobs that require the results from certain correlated jobs. The easy back-filling policy is utilized in the scheduler. In this back-filling technique, the scheduler may back-fill later jobs even if doing so delays the expected start time of other jobs as long as the expected start time of the first job is not affected. The back-fill algorithm for our proposed scheduler is presented in Algorithm 1.

Algorithm for finding the back-fill jobs Find the shadow time and extra nodes: sequences S_i sort the jobs according to their expected finish time. loop over the sequence and collect nodes until the number of available nodes is sufficient for the first job in the queue. record the time t_shadow when the back-filling happens check whether there are more nodes available other than needed by the first queued job, record the number of extra nodes N. Find a back-fill job: sequences S_i loop

on the S_i in order of arrival time t_a each job j_n check whether either of the following conditions hold: it requires no more than the currently free nodes, and will terminate by the shadow time OR it requires no more than the minimum of the currently free nodes and the extra nodes. end for The first such job can be used for back-filling

Scheduler control model

The rule-based control method has been used as the main base for the scheduler design. The architecture of the scheduler is presented in Figure 5.2.

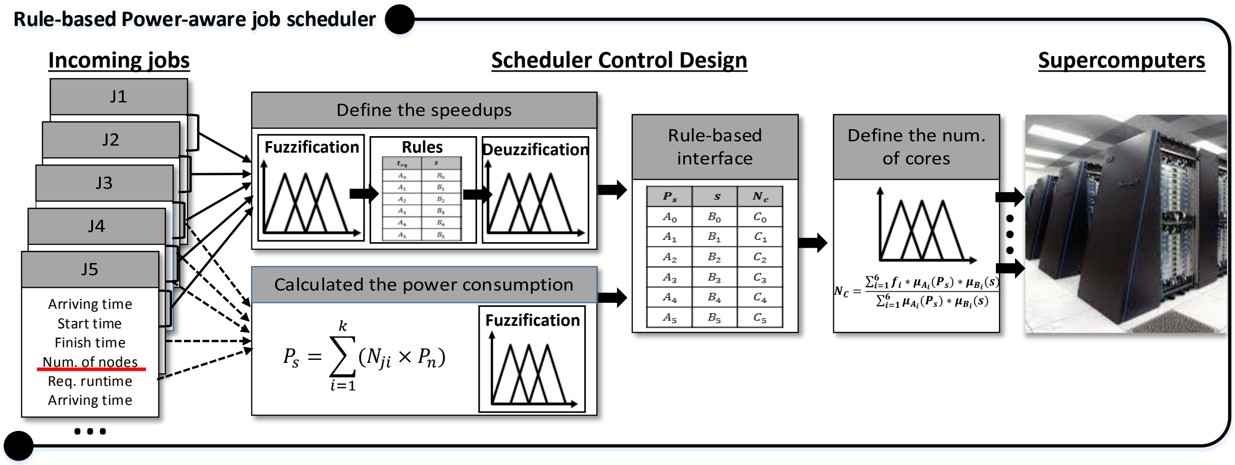


Figure 5.2: The architecture of our Rule-based Power-aware job scheduler

The formulation of the problem is presented using the following equations.

$$P_j = N_j * P_n \tag{5.1}$$

where P_j stands for the power cost per job, P_n stands for the power cost per node. Total power cost for the sequence would be described using the following equation (5.2):

$$P_s = \sum_{i=1}^{k}(P_{j_i}) \tag{5.2}$$

where k represents the number of jobs in a sequence, P_s stands for the total power cost for a sequence. Base on equation (5.1) and equation (5.2), the equation (5.2) can be presented as (5.3):

$$P_s = \sum_{i=1}^{k}(N_{j_i} * P_n) \tag{5.3}$$

For each job, we collect two different performance information. First, we collect the job start time t_s and job finish time t_f, thus the real job processing time t_r can be calculated as (5.4) and the difference between job real run-time and user estimated run-time will be computed use the equation 5.5:

$$t_r = t_f - t_s \tag{5.4}$$

$$t_d = t_r - t_e \tag{5.5}$$

where t_r stands for the real job processing time, t_s and t_f represent the job start and finish time respectively. After calculating the time difference, we need to collect the job requested run-time t_{r_q} as the reference. This is the user estimated job running time that will not be consistent with the real job running time. In addition, in order to ensure the quality of service, the user submitted job should be completed in approximately the expected job finish time. This offers a further opportunity to save energy because a different speedup setting can be chosen to reduce the number of cores for running the specific job to improve the energy efficiency. This will be further discussed in section 5. It is worth mentioning that the job running time cannot being modeled since the available resources are vary continuously and the incoming jobs are random. The time differences of each job are randomized, thus a formula cannot be derived. In this case, we use a rule-based control method to solve this problem. Rule-based control theory [109] is widely used in machine control [110, 111], which has the advantage that the solution to the problem can be cast in terms that human operators can understand, so that their experience can be used in the design of the controller. This approach is extremely useful when the control objective is hard or impossible to model at the current stage.

Define the speedups

Fuzzification based on rules

In order to determine the appropriate running time of a job, we need to collect the difference between the real run-time and the user estimated run-time based on equations 5.4 and 5.5, which are presented in section 5. This aims to allocate the computing resources rationally while satisfying the user's requirements. To apply the rule based methods, we can fuzzify the time difference, t_d, by performing a scale mapping using a membership function to transfer the range of the crisp values into the corresponding universe of discourse. The universe of discourse is the set of linguistic terms in rule based logic, such as "NS","ME" and "PL", which represent "negative short","moderate" and "positive long" respectively. The time difference is one input of the system that is used to determine the performance constraint. We are using the Gaussian membership function presented in equation 5.6 to map crisp values to linguistic terms. The reason that we apply a Gaussian transformation is because it transforms the original values into a normal distribution, which creates a smooth transformation rather than a linear function. On the other hand, since the value of t_d changes frequently, using a Gaussian membership function can provide the ability to detect the fluctuation and response accordingly. The function is shown in equation 5.6 and the plot is illustrated in figure 5.3.

$$\mu_A(t_d) = \exp\left[-\frac{1}{2}\left(\frac{t_d - t_{d_0}}{\sigma}\right)^2\right] \tag{5.6}$$

Where A stands for the rule-based set of t_d which including $\{$NS,NMS,ME,PML,PL$\}$ represent negative short, negative medium short, moderate, positive medium long, positive long. The rule table was presented in table 5.1. For each crisp value of t_d, we can compute a set of μ that can be used in the defuzzification step by applying certain rules.

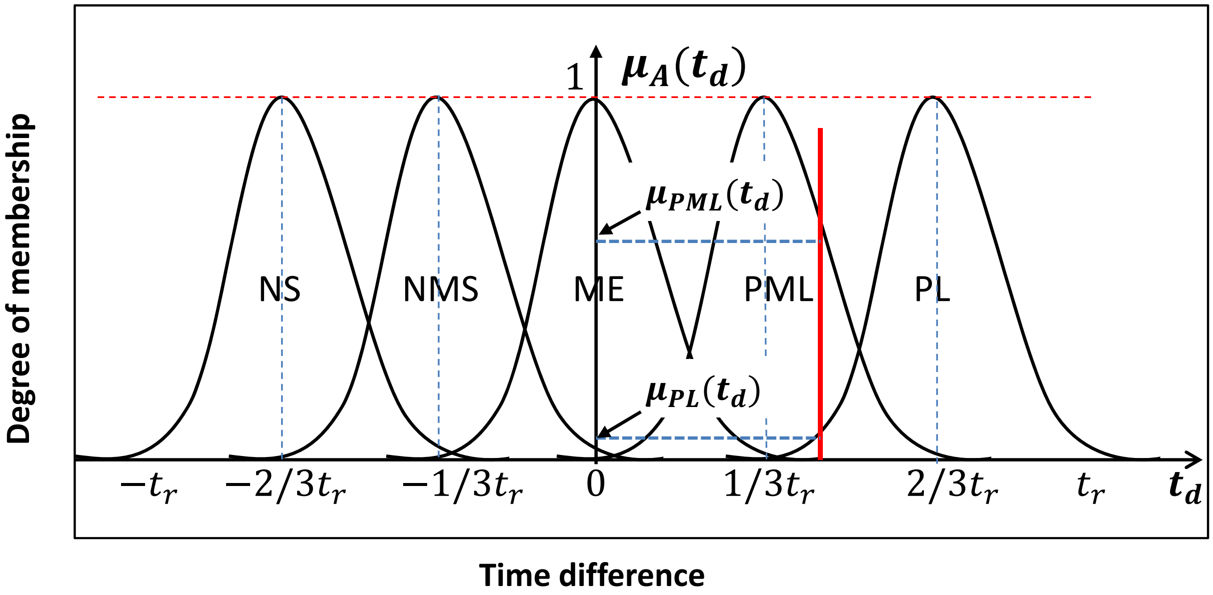


Figure 5.3: Gaussian membership function for mapping the time difference(t_d) to linguistic terms

Table 5.1: Rules for determine the Speedups

INPUT	**OUTPUT**
Time differences	Speedups
PS	S_1
PMS	S_2
ME	S_5
PML	S_4
PL	S_5

Define the power consumption

After categorizing the appropriate run-time of a job, we need to define the power consumption level for each sequence. We define several constraints for the power consumption consumed by the incoming jobs. Specifically, there are 4 power consumption levels associated with 4 different levels of core groups. Basically, we schedule the jobs on either one, two, four, or eight cores.

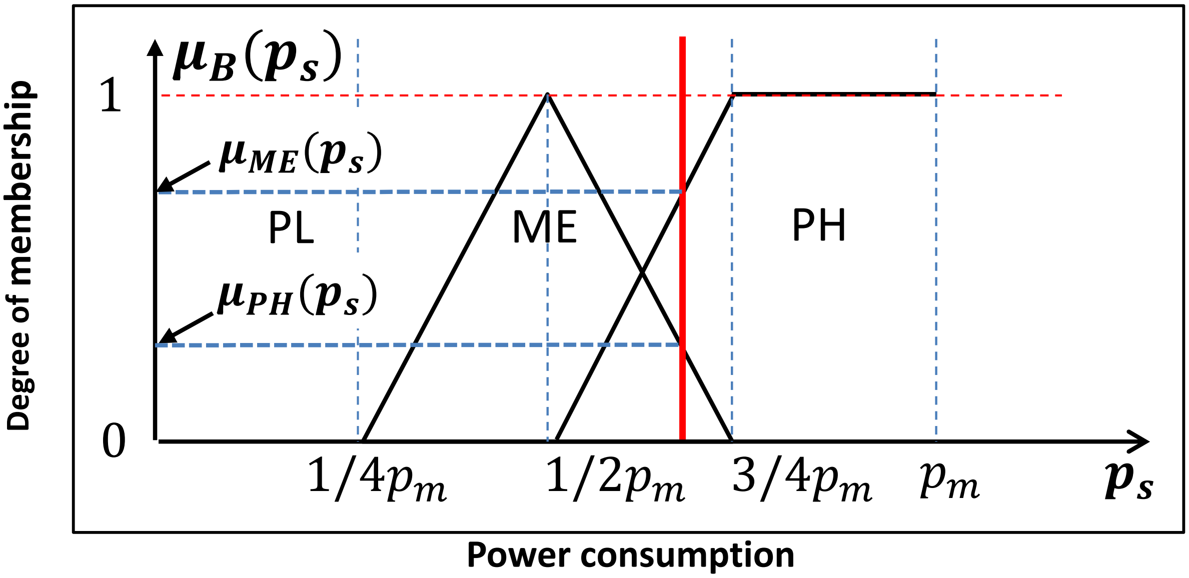


Figure 5.4: Triangular membership function for mapping the power consumption(p_s) to linguistic terms

Since the maximum number of cores for each node is eight, a job scheduled on all 8 cores will consume the most power, which is the upper limit of our defined power constraint. Now, we fuzzify this power input using a triangular membership function presented in equation 5.7, which maps crisp values to linguistic terms. We apply a linear transformation because the power consumption is proportional to the number of active nodes and we have a specified min and max value in this case.

$$\mu_B(p_s) = \begin{cases} 1 - p_s * \delta & : \quad if 0 <= p_s <= p_m \\ 0 & : \quad \text{Otherwise} \end{cases} \tag{5.7}$$

Where δ is a user defined performance loss constraint. p_m denotes the maximum power consumption that can be provided by the supercomputer. B represents the fuzzy set of powers, $\{$PL, ME, PH$\}$ or “positive low”, “moderate” and “positive high”. Figure 5.4 shows an example in which we have a crisp value of power consumption that falls between $1/2p_m$ and $3/4p_m$. We can compute the degree of likelihood under each element of the fuzzy set.

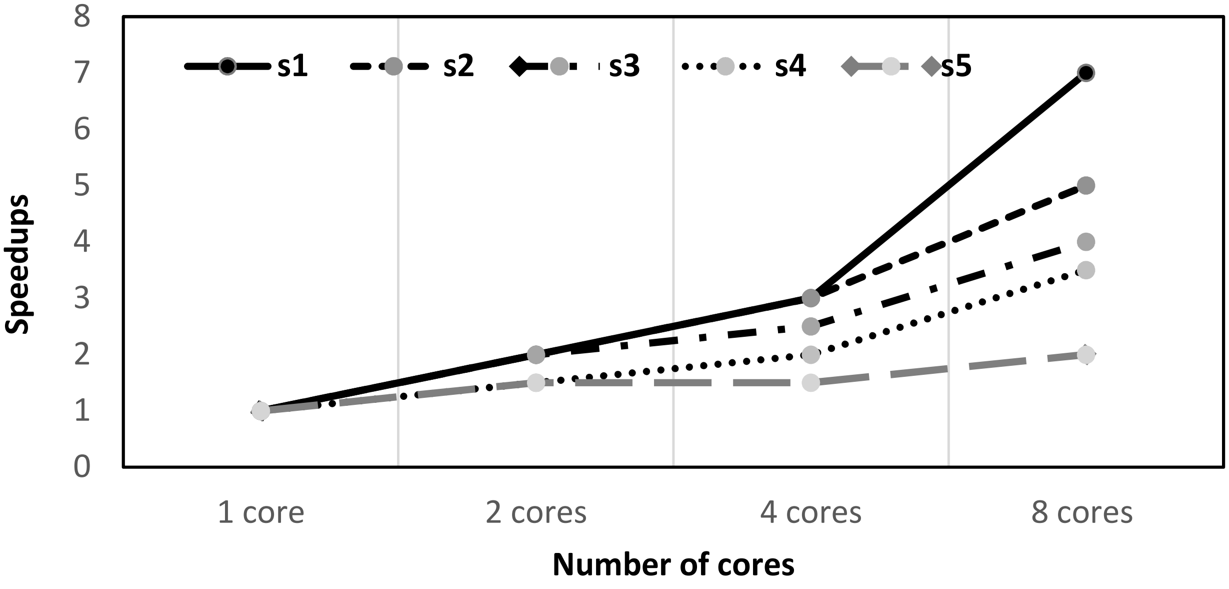


Figure 5.5: The 5 level of speedup profiles

Define the number of cores

After collecting the power and speedup results, we need to decide and allocate the proper number of cores for each job in the queue. The rules for defining the number of cores are listed in table 5.2.

Since we introduce two kinds of membership functions to map the the crisp values t_d and p_s to linguistic terms, there exists two types of centroid defuzzification methods. To determine the speedup, the defuzzification function is presented in 5.8, and the function for determining the number of cores is provided in 5.9.

Table 5.2: Rules for determine of the core numbers

INPUT		**OUTPUT**
Speedups	Power consumption	number of cores
PS	PF	PS
PS	PF	PS
PS	PF	PS
PMS	ME	PS
PMS	ME	PMS
PMS	ME	PMS
ME	PF	PMS
ME	PF	ME
ME	PF	ME
PML	PF	ME
PML	PF	ME
PML	PF	PL
PL	ME	PL
PL	ME	PL
PL	ME	PL

For the determination of speedups:

$$S = \frac{\sum_{i=1}^{5} s_i * \mu_{A_i}(t_d)}{\sum_{i=1}^{5} \mu_{A_i}(t_d)} \tag{5.8}$$

where S stands for speedup results and s_i stands for the 5 speedup levels. For determination of the number of cores:

$$N_c = \frac{\sum_{i=1}^{4} n_i * \sum_{j=1}^{5} \mu_{A_i}(t_d) * \mu_{B_i}(ps)}{\sum_{i=1}^{5} \mu_{A_j}(t_d) * \mu_{B_i}(ps)} \tag{5.9}$$

where N_c stands for the number of cores and n_i represent the 4 level of core selections.

Methodology

In this section, we show our experimental methodology and benchmarks, as well as the implementation details of each component.

Processor

We use an Intel(R)Core(TM)i7-4790 CPU 3.60GHz processor with a 16.0GB Installed memory(RAM). The operating system is 64-bit Windows 7 Enterprise.

Traces

We use four different traces in our experiment. The first trace is from a machine named Blue Horizon at the San Diego Supercomputer Center(denoted as SDSC-BLUE in the paper) [103]. The second trace is the CTC trace, which was obtained from the Feitelson's Parrallel Workloads Archive [104]. The third trace is the ANL Intrepid Log and the forth trace is the Intel Netbatch Grid [106]. The last trace is the Intel Netbatch Logs obtained from the Intel Netbatch Grid [105]. The detailed information of the above traces is listed in the table 5.3. We run each of the above traces utilizing our proposed scheduler and scheduling algorithms.

Power estimation

We suppose that the supercomputer consists of a maximum of 25,600 nodes. The power consumption per node will be 80 watts. In this case, the total power consumption will be 25,600 * 80 watts = 2048,000 watts. We use this number as the hard constraint for making decisions according to the incoming jobs.

Table 5.3: Description of 4 traces

No.	System	Duration	Number of Jobs
1	128-node IBM SP2	05/1998-04/2000	73,496
2	PSC C90 PSC J90 CTC IBM-SP2	01-12/1997 01-12/1997 07/1996-05/1997	54,962 3,582 5,792
3	Blue Gene/P (Intrepid)at ANL	01-09/2009	68,936
4	Intel Netbatch Grid	11/2012	48,821,850

The Hard constraint means that the total power consumption cannot exceed this threshold in any circumstances. In our work, we introduce a dynamic power consumption constraint, which is able to adjust the power threshold to different levels in order to explore further opportunities to improve the power efficiency. We will schedule the job on either one, two , four, or eight cores, where eight cores is the maximum number of cores for each node. In other words, the soft power constraints would be 12.5%, 25%, 50% of the upper limit of the power consumption.

Baseline algorithms

In section 5, we will conduct intensive experiments to compare our scheduler's performance with the baseline algorithms. The baseline algorithms are FIFO(First In First Out), SJF(Shortest Job First), RR(Round Robin), and LJF(Longest Job First). The pros and cons for these algorithms have been presented in table 5.1. FIFO is a commonly used scheduling algorithm which achieves the best user fairness. SJF is able to produce a better system response time and fewer system slowdowns. LJF can achieve the best system utilization but the worst quality of service. Due to the similarity between SJF and LJF scheduling policies, we only choose SJF as the baseline algorithms. RR allocates a time slot or quantum for each job and interrupts the job if it is not finish by then. RR is a preemptive algorithm that results in either max or min fairness depending of the data packet size and provides a guaranteed or differentiated quality of service.

Experiment

In this section, we will present the performance and power conservation results compared with the baseline scheduling algorithms. First, we compare our proposed scheduling algorithm without taking power and speedup profiles into consideration. Secondly, we compare the power efficiency by introducing a new parameter, PPF, which takes power and speedup profiles into account.

Comparison of scheduling algorithm

Scheduling efficiency

In this section, we will compare our proposed scheduling policy and control method to traditional scheduling approaches. Figure 5.6 shows the results of the average system utilization through different scheduling algorithms. In this circumstance, we did not take power and speedup profiles into consideration. This is so we can solely test the performance of the scheduling algorithms. We can see from figure 5.6 that our proposed scheduling algorithm achieves the best system utilization, which is 75% of the total system usage. The overall system utilization of our propose scheduling algorithm outperforms FIFO, SJF, RR by 12%, 23% and 17% respectively. From figure 5.7, we can see that the average response time for our proposed method is 3215.7 seconds, which is about 25.4%, 16.8% and 25% less than FIFO, SJF, and RR respectively. This is because we take advantage of the FIFO scheduling policy to attain a better quality of service, the SJF policy to attain better average response time and we also take advantage of easy back-filling algorithm to achieve a better system utilization.

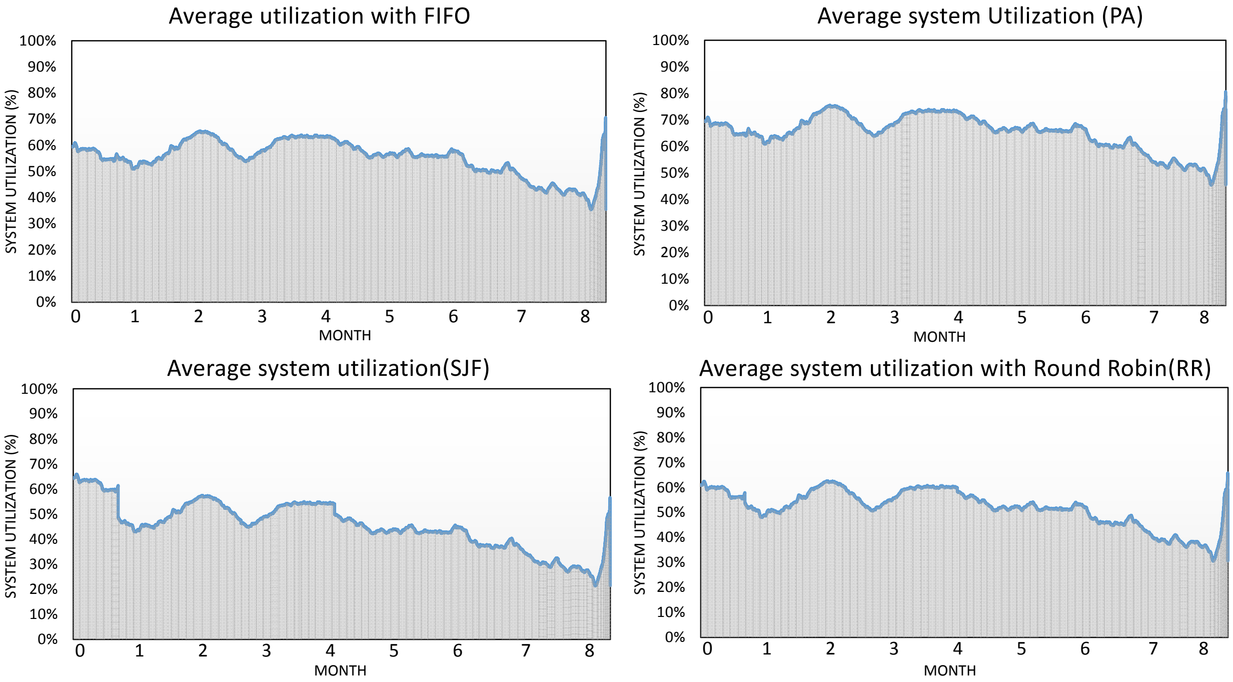


Figure 5.6: Average system utilization compared with the baseline scheduling polices using trace 3

Determine the length of a Sequence

Defining the length of the sequence is critical for our proposed job scheduling algorithm, where a small difference can result in poor use of substantial resources. For trace 3, presented in table 5.3, we run the scheduling algorithm repeatedly with different lengths of sequences to examine the relationship between scheduling performance and sequence length as well as to find the proper length of sequence for a specific case. Figure 5.9 shows the performance output in terms of system utilization and average response time with sequences of different lengths. We are able to achieve the best performance result when the L_s is equal to 100.

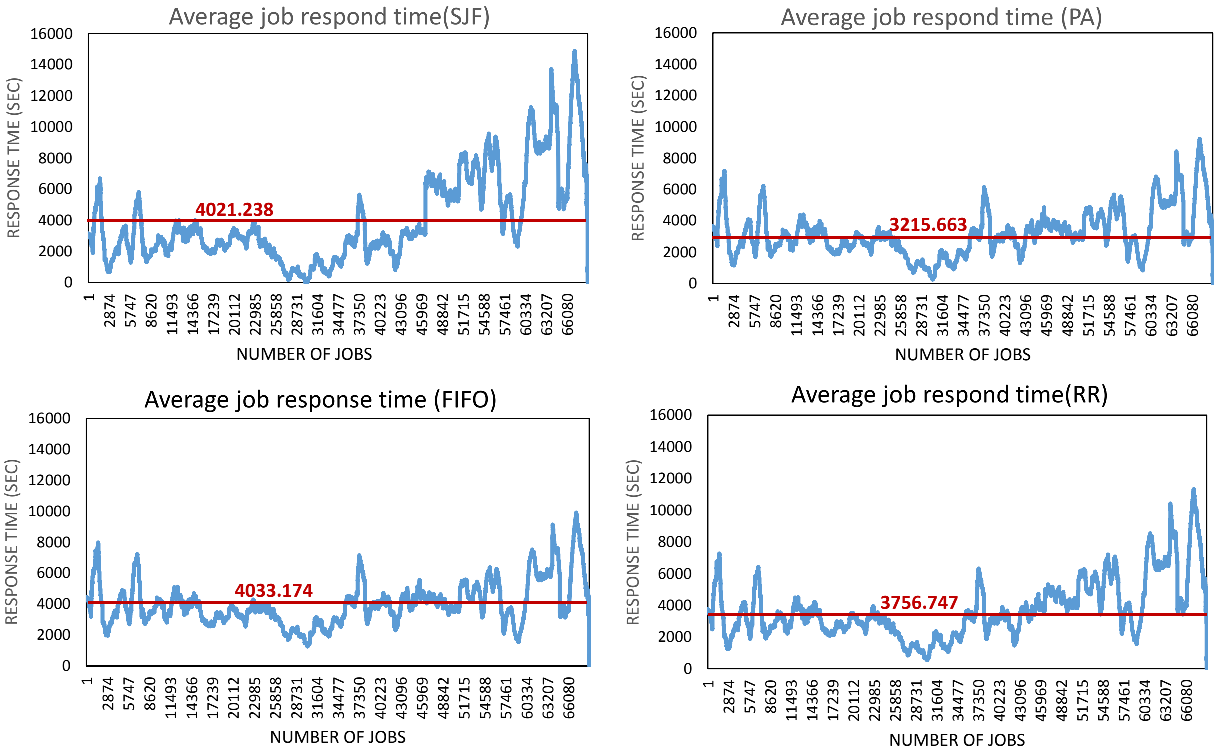


Figure 5.7: Average system response time compared with the baseline scheduling polices using trace 3

Power and Speedup constraints

In this section, we will take power and speedup profiles into account to evaluate the power efficiency of our proposed scheduler. Normally, we understand that the power consumption is inversely proportional to the overall performance. This means that consuming more power will lead to a better performance, e.g. shorter average response times. This relationship works well if there are no restrictions on the amount of power consumption. However, in our case, we will fix the total energy consumption due to the power-capping requirement. Thus, we cannot use the previous power performance relationship to evaluate our proposed the method. Therefore, we propose a new power performance relationship called the “Power-Performance-Factor”, denoted as PPF.

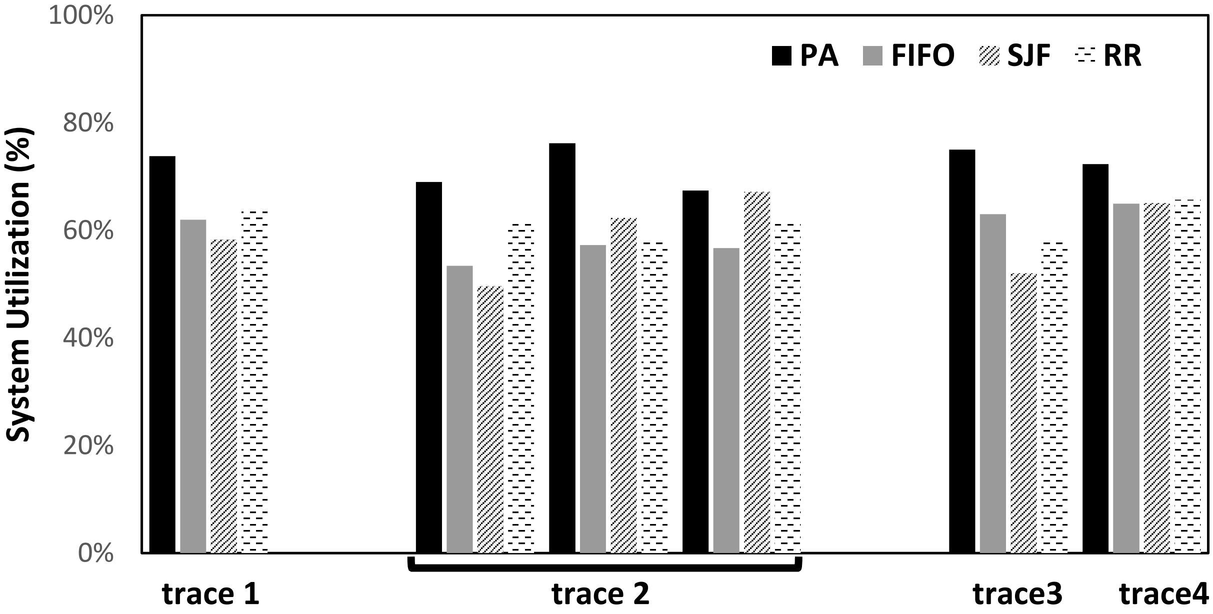


Figure 5.8: Average system utilization for all 4 traces

This factor takes into consideration both the power and performance as one parameter to evaluate the power efficiency under the pre-defined power constraints. The original PPF factor is obtained by multiplying the power consumption with its relevant response time of the jobs scheduled on all 8 cores of each node. This $PPF_{Original}$ implies the maximum power consumption and minimum average response time. Intuitively, if the power consumption is reduced, the average response time will increase accordingly. However, if the ratio between the new PPF factor and the $PPF_{Original}$ is less than 1, this indicates that the power efficiency is improved. In other words, the reduced power only slightly affect the average response time, but does not have a substantial influence on the quality of service.

First, we run our power-aware scheduling algorithm using trace 3 as the input introduced in table 5.2. Figure 5.11 and 5.12 shows the scheduling output with and without taking energy and speedup into considerations.

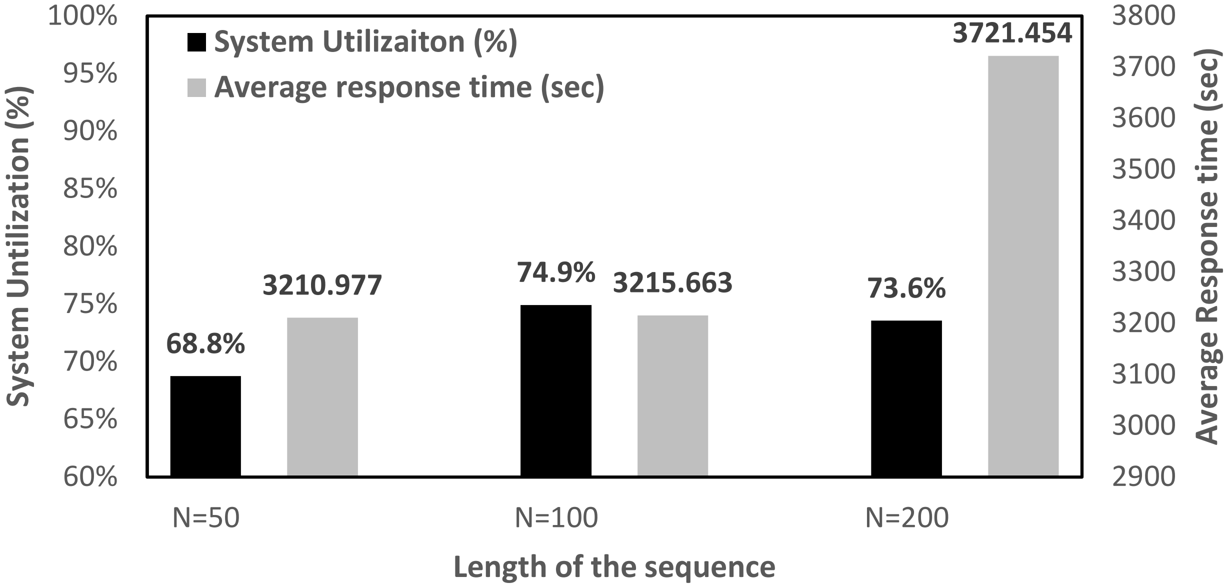


Figure 5.9: System utilization and response time with different settings of the length of Sequence (L_s) by using trace 3

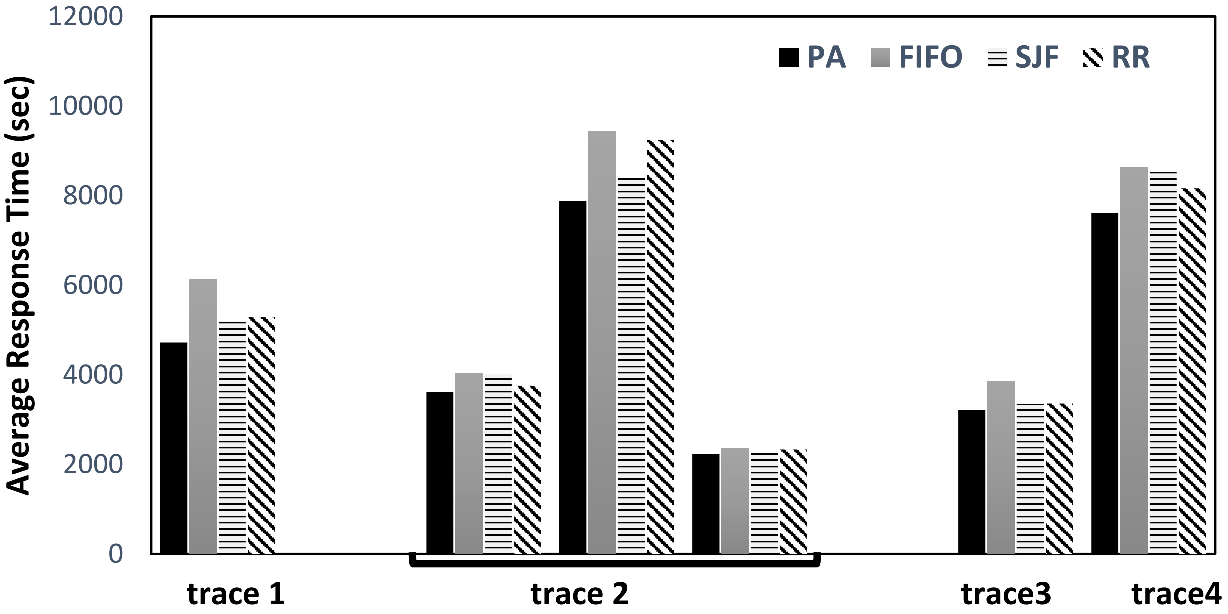


Figure 5.10: Average response time for all 4 traces

We can find from Figure 5.11 that if we schedule the jobs on one core of each available node, we could increase the job run-time. However, the power performance factor will be reduced to 0.25 of the original PPF factor. In other words, we increase the power efficiency by 75%. Figure 5.12 presents another case in power constraint together and its corresponding run-time performance with and without taking power and speedup profiles into consideration. The job has been scheduled on 2,4,8 cores with the percentage of 20%, 50% and 30%. We observe that the average power consumption will be 1228.8KW and the PPF factor is 0.75 of PPF original. This shows that the power efficiency has been increased by 25%. Secondly, we run all 4 traces while considering the power and speedup profiles and present the results in Figure 5.14. We can see that the average power consumption for the four traces are 441.75 KW,653.95KW, 102.2KW,59.33KW, 512KW, and 1925.175KW. The breakdowns of the scheduled percentage on various core groups for each trace are shows in table 5.4. The corresponding PPF factors are 0.64,0.72,0.95,0.88,0.25 and 0.74 with a power efficiency improvement of 36%,28%,5%,12%,75% and 26% respectively.

Table 5.4: The breakdowns of the scheduling percentage on different core groups

Trace No.	**1 core**	**2 cores**	**4 cores**	**8 cores**
1	0%	50%	30%	20%
2	10% 0% 15%	20% 20% 25%	50% 30% 10%	20% 50% 50%
3	100%	0%	0%	0%
4	15%	25%	10%	50%

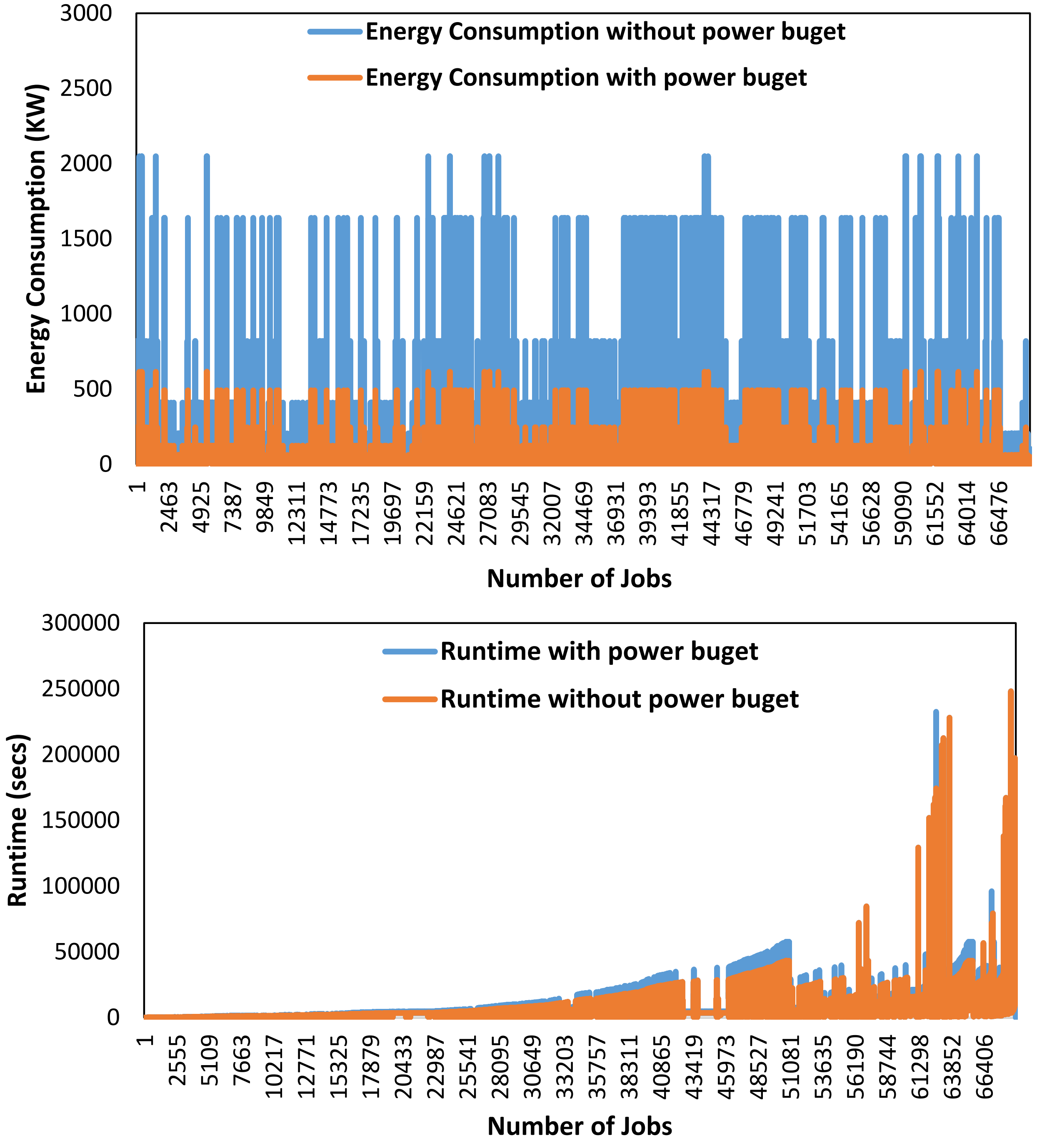


Figure 5.11: Power consumption results with considering of the power constraints, P = 512KW

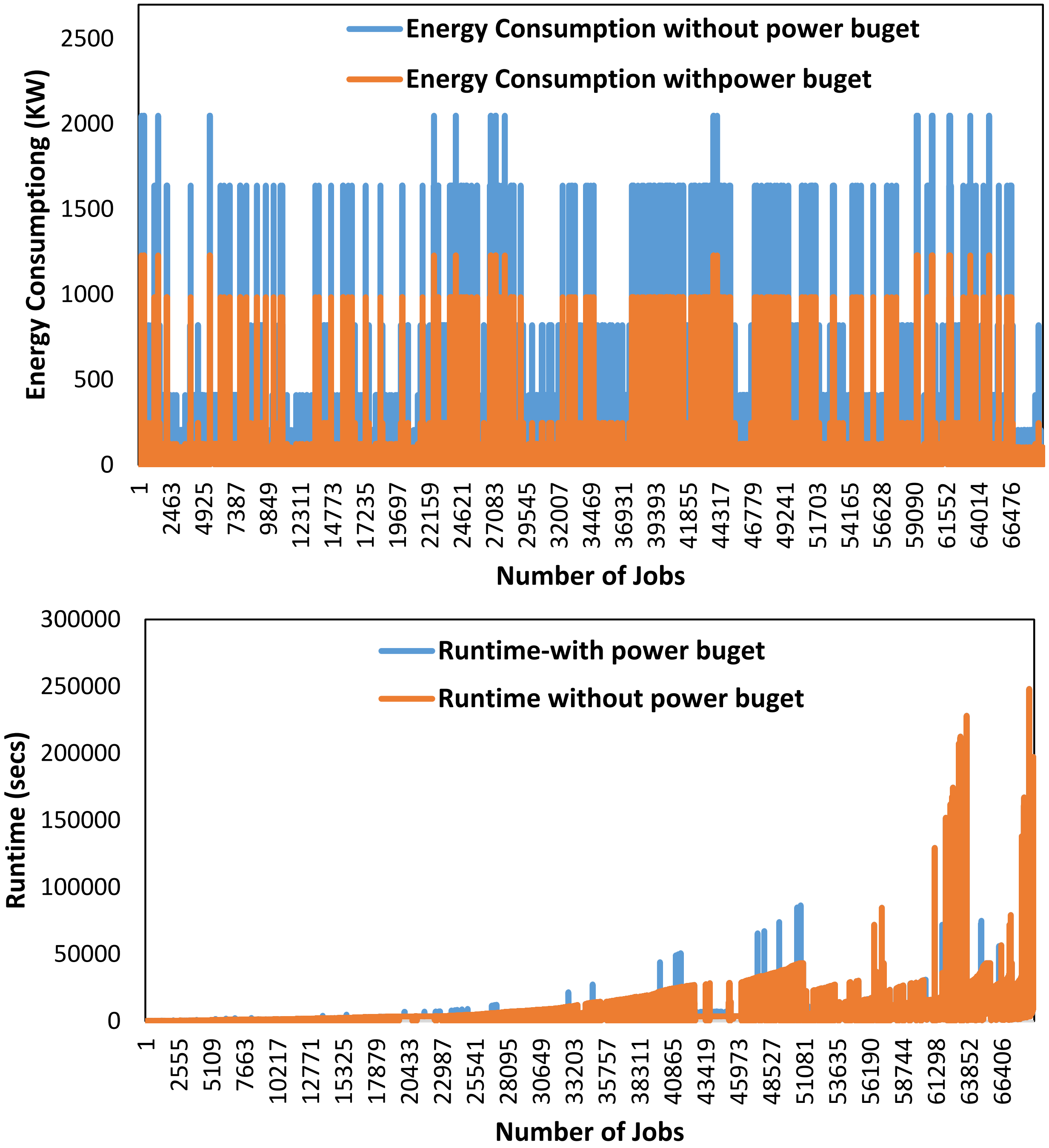


Figure 5.12: Power consumption results with considering of the power constraints, P = 1228.8KW

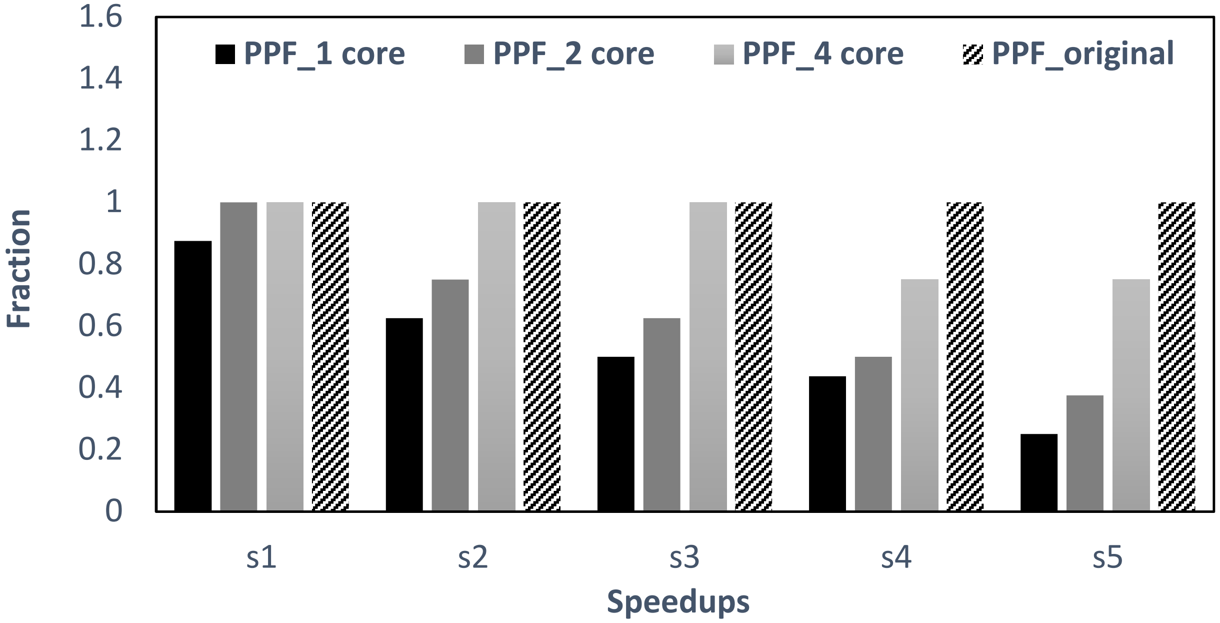


Figure 5.13: The result of Power Performance Factor for 5 different speedup levels by using trace 3

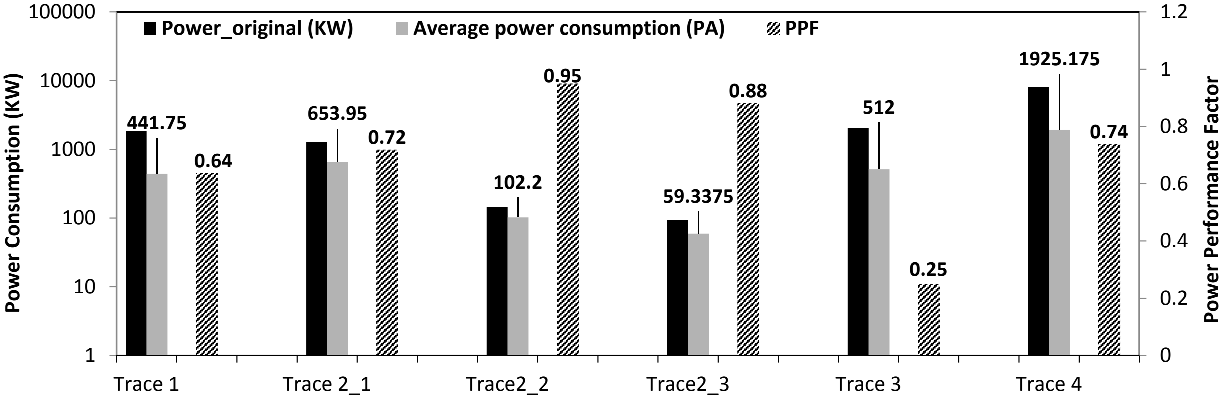


Figure 5.14: Average power consumption and PPF output for all 4 traces

CHAPTER 6: CONCLUSION

In this section, we present concluding remarks of the proposed designs.

Firstly, Power control for multi-core systems has become increasingly important and challenging. However, existing power control solutions cannot be directly applied into CMP systems because of the new data-intensive applications and complicate job scheduling strategies in CMP systems. In this work, we present MAR, a model-free, adaptive, rule-based power management scheme in multi-core systems to manage the power consumption while maintain the required performance. "Model-less" reduces the complexity of system modeling as well as the risk of design errors caused by statistical inaccuracies or inappropriate approximations. "Adaptive" allows MAR to adjust the control methods based on the real-time system behaviors. The rules in MAR are derived from experimental observations and operators' experience, which create a more accurate and practical way to describe the system behavior. "Rule-based" architecture also reduces the development cycle and control overhead, simplifies design complexity. MAR controller is highly responsive (including short detective time and settling time) to the workload bouncing by incorporating more comprehensive control references (e.g. changing speed, I/O wait). Empirical results on a physical testbed show that our control method could achieve more precise power control result as well as higher power efficiency for optimized system performance compared to other four existing solutions. Based on our comprehensive experiments, MAR could outperform the baseline methods by 12.3%-16.1% in power saving efficiency, and maintains comparable performance loss about 0.78%-1.08%. In our future research, we will consider applying fuzzy logic control in power conservation of storage systems.

Secondly, we propose a new reliability model to analyze the reliability of multi-way replication storage systems with different data layout schemes. We make the following conclusions:

- Modeling of the system reliability should not only taking data loss probability and recovery bandwidth into account, but also need to considering about the probability of replica lost in order to obtain an accurate result.

- The reliability of random declustering layout is highly dependent on the number of redundancy groups in the system. With the increase of redundancy groups and/or the number of disks in the system, the reliability of random declustering drops.

- The shifted declustering is less sensitive to the scale of the storage system comparing with random declustering. With the same resource provided for recovery per disk, the shifted declustering layout achieves almost 100% reliable that lasting for 10-years long period. In particular, the data integrity of the shifted declustering layout lasts 85% times longer in our simulation than random declustering layout.

- The Copyset replication method obtained the best system reliability due to its carefully arrangement of the data blocks. However, with considering of the recovery bandwidth, the system reliability has been greatly affected especially when the bandwidth is low.

- Our study on both 5-year and 10-year system reliability equipped with various recovery bandwidth settings shows that, the shifted declustering layout surpasses the two baseline approaches in both cases by consuming up to 83 % and 97% less recovery bandwidth for copyset, as well as 5.2% and 11% less recovery bandwidth for random layout.

As random declustering layout is widely adopted by enterprise large-scale systems configured with multi-way replication, shifted declustering layout is a promising alternative for its proved optimal performance and high reliability with low recovery overhead.

Finally, we develop a power-aware job scheduler by applying a rule based control method and taking into consideration real world power and speedup profiles to improve power efficiency while adhering to predetermined power constraints.Power conservation is increasingly critical nowadays for large scaled storage systems and supercomputers since the annual electricity expenditure is extremely expensive. Many researchers have focused on reducing the energy consumption by incorporating the traditional DVFS technique into their specific methods depending on the real workloads. However, this method is limited in terms of energy savings, especially when the workload is heavy. Therefore, we develop a rule-based power-aware job scheduler, which takes both power and speedup profiles into consideration to improve the power efficiency of the supercomputer regardless of how intensive the workload is. Specifically, the classic relationship between power consumption and performance is not suitable for this circumstance since a power constraint is introduced. Thus, we propose a new parameter named PPF to evaluate the power efficiency with various power and speedup settings. A rule-based scheduler is utilized because the time difference input varies continuously and is randomized and thus cannot be represented by a formula. Finally, our proposed method is able to increase the power efficiency by up to 75%.

LIST OF REFERENCES

[1] V. Hanumaiah, R. Rao, S. Vrudhula, and K. S. Chatha, *Throughput opti-mal task allocation under thermal constraints for multi-core processors,* in DAC 09: Proceedings of the 46th Annual Design Automation Conference. New York, NY, USA: ACM, 2009, pp. 776781.

[2] S. R. Alam, P. K. Agarwal, and J. S. Vetter, *Performance characteristics of bio-molecular simulations on high-end systems with multi-core processors,* Parallel Compute, vol. 34, no. 11, pp. 640651, 2008.

[3] C. Isci, A. Buyuktosunoglu, C. yong Cher, P. Bose, and M. Martonosi, *An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget,* in Proc. Intl Symp. Microarch. (MICRO), 2006, pp. 347358.

[4] A. Flores, J. L. Aragon, and M. E. Acacio, *An energy consumption charac-terization of on-chip interconnection networks for tiled CMP architectures,* J. Supercomput., vol. 45, no. 3, pp.341364, 2008.

[5] *Introducing the 45nm next-generation intel core microarchitecture,* 2007.[Online].Available:http://www.intel.com/technology/architecture-silicon/intel64/45nm-core2 whitepaper.pdf

[6] *Amd coolnquiet technology,* 2006. [Online]. Available: http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx

[7] *IBM energyscale for power6 processor-based systems,* 2009. [Online]. Available:http://www 03.ibm.com/systems/power/hardware/whitepapers/energyscale.html

[8] Q. Wu, V. J. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. W. Clark, *A dynamic compilation framework for controlling microprocessor energy and performance,*

in MICRO 38: Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Soci-ety, 2005, pp. 271282.

[9] C. Isci, G. Contreras, and M. Martonosi, *Live, runtime phase monitoring and prediction on real systems with application to dynamic power management,* in MICRO 39: Proceedings of the 39th Annual IEEE/ACM Interna-tional Symposium on Microarchitecture. Washington, DC, USA: IEEE Com-puter Society, 2006, pp. 359370.

[10] A. Weissel and F. Bellosa, *Process cruise control: event-driven clock scaling for dynamic power management,* in CASES 02: Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems. New York, NY, USA: ACM, 2002, pp. 238246.

[11] F. Xie, M. Martonosi, and S. Malik, *Bounds on power savings using runtime dynamic voltage scaling: an exact algorithm and a linear-time heuristic approximation,* in ISLPED 05: Proceedings of the 2005 international symposium on Low power electronics and design. New York, NY, USA: ACM, 2005, pp. 287292.

[12] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, *An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget,* in MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Micro-architecture.Washington, DC, USA: IEEE Computer Society, 2006, pp. 347358.

[13] R. Teodorescu and J. Torrellas, *Variation-aware application scheduling and power management for chip multiprocessors,* in ISCA 08: Proceedings of the 35th International Symposium on Computer Architecture. Washington, DC, USA: IEEE Computer Society, 2008, pp. 363374.

[14] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, *Managing server energy and operational costs in hosting centers,* in SIGMETRICS 05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. New York, NY, USA: ACM, 2005, pp. 303314.

[15] *Cluster-level feedback power control for performance optimization,* IEEE Computer Society, 2008.

[16] Y.Wang, K. Ma, and X. Wang, *Temperature-constrained power control for chip multiprocessors with online model estimation,* in ISCA 09: Proceedings of the 36th annual international symposium on Computer architecture. New York, NY, USA: ACM, 2009, pp. 314324.

[17] A. Fedorova, M. Seltzer, and M. D. Smith, *Improving performance isolation on chip multiprocessors via an operating system scheduler,* in PACT 07: Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques. Washington, DC, USA: IEEE Computer Society, 2007, pp. 2538.

[18] *Processor utilization and wait I/O a look inside,* 2008. [Online]. Available:http://sunsite.uakom.sk/sunworldonline/swol-08-1997/swol-08-insidesolaris.html

[19] *Enhanced intel speedstep technology for the intel pentium m processor,* 2004. [Online]. Available: http://xnu-speedstep.googlecode.com/files/PentiumM SpeedStepDoc.pdf

[20] M. Beck, R. Magnus, and U. Kunitz, *Linux Kernel Internals with Cdrom,* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[21] D. Bovet and M. Cesati, *Understanding the Linux Kernel.* Oreilly Associ-ates Inc, 2005.

[22] *Isolation benchmark suite,* 2007. [Online]. Available: http://web2.clarkson.edu/class/cs644/isolation/

[23] Spec cpu2006, 2006. [Online]. Available: http://www.spec.org/cpu2006/

[24] *Tpcc-uva: A free, open-source implementation of the TPC-C benchmark*, 2006. [Online]. Available: http://www.infor.uva. es/diego=tpcc □ uva:html

[25] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, *CPU miser: A perfor-mance-directed, run-time system for power-aware clusters*, in ICPP 07: Proceedings of the 2007 International Conference on Parallel Processing. Washing-ton, DC, USA: IEEE Compute Society, 2007, p. 18.

[26] V. Vishwanath, R. Burns, J. Leigh, and M. Seablom, *Accelerating tropical cyclone analysisusing lambdaram, a distributed data cache over wide-area ultra-fast networks*, FutureGener. Comput. Syst., vol. 25, no. 2, pp. 184191, 2009.

[27] M. Kandemir, S. W. Son, and M. Karakoy, *Improving i/o performance of applications through compiler-directed code restructuring*, in FAST08: Proceedings of the 6th USENIX Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX Association, 2008, pp. 116.

[28] S. W. Son, G. Chen, M. Kandemir, and F. Li, *Energy savings through em-bedded processing on disk system,* in ASP-DAC06: Proceedings of the 2006 Asia and South Pacific Design Automation Conference. Piscataway, NJ, USA: IEEE Press, 2006, pp. 128-133.

[29] Is i/o wait a measure of cpu utilization or idle time? 2005. [Online]. Available: http://www.ee.pw.edu.pl/pileckip=aix=waitio2:htm

[30] K. Elmeleegy, A. Chanda, A. L. Cox, and W. Zwaenepoel, *Lazy asynchronous i/o for event-driven servers*, in ATEC 04: Proceedings of the annual conference on USENIX Annual Technical Conference Berkeley, CA, USA: USENIX Association, 2004, pp. 2121.

[31] K.-D. K. X. L. Can Basaran, Mehmet H. Suzer, *Model-free fuzzy control of CPU utilization for unpredictable workloads,* in the Fourth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks, 2009.

[32] B. G. A. Varma, *A control-theoretic approach to dynamic voltage scheduling*, in Proc. International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASE 2003), San Jose CA, 2003.

[33] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, *Ship: Scalable hierarchical power control for large-scale data centers*, in PACT 09: Proceedings of the 2009 18th International Conference on Parallel Architectures and Compilation Techniques. Washington, DC, USA: IEEE Computer Society, 2009, pp. 91100.

[34] Rtai - the realtime application interface for linux, 2010. [Online]. Available: https://www.rtai.org/

[35] C. Lu, X. Wang, and X. Koutsoukos, *Feedback utilization control in dis-tributed real-time systems with end-to-end tasks*, IEEE Trans. Parallel Dis-trib. Syst., vol. 16, no. 6, pp. 550-561, 2005.

[36] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos, *SESC simulator*, January 2005, http://sesc.sourceforge.net.

[37] O.N. Ec-Rjrza-Te, Alpha 21264 microprocessor hardware reference manual.

[38] Hardware performance counters basics. [Online]. Available: http://perfsuite.ncsa.uiuc.edu/publications/LJ135/x27.html

[39] X. Wang, D. Jia, C. Lu, and X. Koutsoukos, *Deucon: Decentralized end-to-end utilization control for distributed real-time systems*, IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 7, pp. 9961009, 2007.

[40] L.-X.Wang, A course in fuzzy systems and control. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[41] T. Shaocheng, L. Changying, and L. Yongming, *Fuzzy adaptive observer backstepping control for mimo nonlinear systems*, Fuzzy Sets Syst., vol. 160, no. 19, pp. 27552775, 2009.

[42] S. Tong and Y. Li, *Observer-based fuzzy adaptive control for strict-feedback nonlinear systems*, Fuzzy Sets Syst., vol. 160, no. 12, pp. 17491764, 2009.

[43] Why use fuzzy logic? 2000. [Online]. Available: http: //www.aptronix.com/fide/whyfuzzy.htm

[44] L.A. Zadeh, Is there a need for fuzzy logic? Information Science, vol.178, pp.2751-2779, 2008. [Online]. Available: http://www.sciencedirect.com

[45] M. Bansal, D. Pushpan, H. Medha, S. Jain, and S. Jain, *CPU frequency scaling by utilization*, Technical report, 2012.

[46] A. Cidon, S. Rumble, R. Stutsman, S. Katti, J. Ousterhout and M. Rosenblum Copysets: reducing the frequency of data loss in cloud storage, Presented as part of the 2013 USENIX Annual Technical Conference, pages 37–48, 2013

[47] Cisco Nexus 7000 Series 32-port 10Gb Ethernet Module, 80Gb Fabric. http://www.cisco.com/en/US/prod/collateral /switches/ps9441/ps9402/ps9512/Data SheetC78-437757.html, 2008.

[48] Hadoop project. http://hadoop.apache.org/core/, 2008.

[49] Inclusion-exclusion principle. http://mathworld.wolfram.com /Inclusion-ExclusionPrinciple.html, 2008.

[50] F.Dinu, and T.S. Eugene Ng, *Understanding the Effects and Implications of Compute Node Related Failures in Hadoop*, Proceedings of the High Performance Parallel and Distributed Computing (HPDC), pages 187-197, 2012.

[51] Q. Xin, E.L. Miller, and T.J.E. Schwarz, *Evaluation of distributed recovery in large-scale storage systems*, Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing, pages 172-181, 2004.

[52] Product Manual Barracuda 7200.11 Serial ATA.

http://www.seagate.com/staticfiles/support/disc/manuals /desktop/Barracuda%207200.11 /100452348e.pdf, 2008.

[53] Seagate Technology - Barracuda 7200.11 SATA 3GB/s 1.5-GB Hard Drive. http://www.seagate.com/ww /v/index.jsp?vgnextoid=511a8cf6a794b110VgnVCM100000f 5ee0a0aRCRD&locale=en-US&reqPage=Model &modelReqTab=Features, 2008.

[54] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. *Farsite: Federated, available, and reliable storage for an incompletely trusted environment.* In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), pages 1–14, 2002.

[55] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. *An analysis of latent sector errors in disk drives.* SIGMETRICS Perform. Eval. Rev., 35(1):289–300, 2007.

[56] R. E. Bryant. *Data-intensive supercomputing: The case for DISC*. Technical Report CMU-CS-07-128, Carnegie Mellon University, May 2007.

[57] M.-S. Chen, H.-I. Hsiao, C.-S. Li, and P. S. Yu. *Using rotational mirrored declustering for replica placement in a disk-array-based video server*. Multimedia Syst., 5(6):371–379, 1997.

[58] S. Chen and D. Towsley. *A performance evaluation of RAID architectures.* IEEE Trans. Comput., 45(10):1116–1130, 1996.

[59] G. Copeland and T. Keller. A *comparison of high-availability media recovery techniques*. In SIGMOD ’89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data, pages 98–109, New York, NY, USA, 1989. ACM.

[60] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. *Dynamo: amazon’s highly available key*

value store. In SOSP'07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, pages 205–220, New York, NY, USA, 2007. ACM.

[61] J. Gafsi and E. W. Biersack. *Modeling and performance comparison of reliability strategies for distributed video servers*. IEEE Transactions on Parallel and Distributed Systems, 11:412–430, 2000.

[62] S. Ghemawat, H. Gobioff, and S.-T. Leung. *The google file system*. SIGOPS Oper. Syst. Rev., 37(5):29–43, December 2003.

[63] A. Gulati, A. Merchant, and P. J. Varman. *pclock: an arrival curve based approach for qos guarantees in shared storage systems*. SIGMETRICS Perform. Eval. Rev., 35(1):13–24, 2007.

[64] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 4th edition, 2006.

[65] M. Holland and G. A. Gibson. *Parity declustering for continuous operation in redundant disk arrays*. In ASPLOS, pages 23–35, 1992.

[66] R. Honicky and E. Miller. *Replication under scalable hashing: a family of algorithms for scalable decentralized data distribution*. In Proceedings of 18th International Parallel and Distributed Processing Symposium, 2004., page 96, 26-30 April 2004.

[67] H.-I. Hsiao and D. J. DeWitt. *Chained declustering: A new availability strategy for multiprocessor database machines*. In Proceedings of the Sixth International Conference on Data Engineering, pages 456–465, Washington, DC, USA, 1990. IEEE Computer Society.

[68] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. *Random sampling techniques for space efficient online computation of order statistics of large datasets*. In SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data, pages 251–262, New York, NY, USA, 1999. ACM.

[69] The MathWorks. Matlab. http://www.mathworks.com.

[70] R. Nelson. Probability, Stochastic Processes, and Queueing Theory. Springer-Verlag, 1995.

[71] R. Sahnar, K. S. Trived, and A. Puliafito. Sharpe. http://www.ee.duke.edu/ kst/software packages.html, 2001.

[72] B. Schroeder and G. A. Gibson. *Understanding disk failure rates: What does an mttf of 1,000,000 hours mean to you?* Trans. Storage, 3(3):8, 2007.

[73] A. Thomasian and M. Blaum. *Mirrored disk organization reliability analysis*. IEEE Transactions on Computers, 55(12):1640–1644, 2006.

[74] A. S. Tosun. Analysis *and comparison of replicated declustering schemes*. IEEE Trans. Parallel Distrib. Syst., 18(11):1578–1591, 2007.

[75] K. S. Trivedi. *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice-Hall, 1982.

[76] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou. *Scalable performance of the panasas parallel file system*. In FAST'08: Proceedings of the 6th USENIX Conference on File and Storage Technologies, pages 1–17, Berkeley, CA, USA, 2008. USENIX Association.

[77] J. M. Wing. *Computer science meets science and engineering*. HEC FSIO R&D Workshop, NSF, August 2007.

[78] Q. Xin. *Understanding and coping with failures in large-scale storage systems*. Technical,Report UCSC-SSRC-07-06, Storage Systems Research Center, Baskin School of Engineering, University of California, Santa Cruz, May 2007. This Ph.D. thesis was originally filed in December, 2005.

[79] Q. Xin, E. L. Miller, T. Schwarz, D. D. E. Long, S. A. Brandt, and W. Litwin. *Reliability mechanisms for very large storage systems*. In MSS '03: Proceedings of the 20 th IEEE/11 -th NASA Goddard Conference on Mass Storage Systems and Technologies (MSS'03), page 146,Washington, DC, USA, 2003. IEEE Computer Society.

[80] H. Zhu, P. Gu, and J. Wang. *Shifted Declustering: a placement-ideal layout scheme for multiway replication storage architecture*. In ACM International Conference on Supercomputing, pages 134–144, 2008.

[81] J.Wang, R.Wang, J.Yin, H. Zhu, and Y.Yang. *Reliability Analysis on Shifted and Random Declustering Block Layouts in Scale-out storage Architectures*. In IEEE International Conference on Networking, Architecture and Storage, Tianjing, 2014.

[82] A.A. Chandio, K. Bilal, N. Tziritas, Z. Yu, Q. Jiang, S.U. Khan and C.Z Xu. *A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems*. Cluster Computing, vol. 17, no.4, pages 1349–1367,2014. Springer.

[83] S. Soner and C. Ozturan. *An auction based slurm scheduler for heterogeneous supercomputer and its comparative performance study*. Technical report, PRACE, 2013. http://www. prace-project. eu/IMG/pdf/wp59 an auction based slurm scheduler heterogeneous supercomputers and its comparative study. pdf, 2013.

[84] Y. Wang and P. Lu. *DDS: A deadlock detection-based scheduling algorithm for workflow computations in HPC systems with storage constraints.* Parallel Computing, vol.39, no.8, pages 291–305, 2013. Elsevier.

[85] C. Liu, X. Qin, S. Kulkarni, C. Wang, S. Li, A. Manzanares and S. Baskiyar. *Distributed energy efficient scheduling for data-intensive applications with deadline constraints on data grids.* IEEE International Conference on Performance, Computing and Communications Conference, pages 26–33, 2008.

[86] S. Niu, J. Zhai, X. Ma, M. Liu, Y. Zhai,W. Chen andW. Zheng. *Employing checkpoint to improve job scheduling in large-scale systems.* Job Scheduling Strategies for Parallel Processing, pages 36–55, 2013. Springer.

[87] G. Varsamopoulos, A. Banerjee and S. K. Gupta. *Energy efficiency of thermal-aware job scheduling algorithms under various cooling models.* Contemporary Computing, pages 568–580, 2009. Springer.

[88] Y. Zhang, L. Duan, B. Li, L. Peng and S. Sadagopan. *Energy efficient job scheduling in single-ISA heterogeneous chip-multiprocessors.* 15th IEEE International Symposium on Quality Electronic Design (ISQED), pages 660-666, 2014.

[89] H. L. Chan, W. Chan, T.W. Lam, L.K. Lee, K.S. Mak and P.W. Wong. *Energy efficient online deadline scheduling*. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pages 795–804, 2013.

[90] K. Fox, S. Im and B. Moseley. *Energy efficient scheduling of parallelizable jobs*. Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 948–957, 2013.

[91] S. Khuller, J. Li and B. Saha. *Energy efficient scheduling via partial shutdown*. Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, pages 1360–1372, 2010.

[92] O. Mammela, M. Majanen, R. Basmadjian, H. DeMeer, A. Giesler andW. Homberg. *Energy-aware job scheduler for high-performance computing*. Computer Science-Research and Development, vol.27, no.4, pages 265–275, 2012. Springer.

[93] S.S. Deore, and A.N. Patil. *Energy-efficient job scheduling and allocation scheme for virtual machines in private clouds*. Journal of Energy, vol.5, no.1, 2013.Citeseer.

[94] X. Wang, Y. Wang and H. Zhu. *Energy-efficient task scheduling model based on MapReduce for cloud computing using genetic algorithm*. Journal of Computers, vol.7, no.12, pages 2962–2970,

[95] J.J. Chen and T.W. Kuo. *Energy-efficient scheduling of periodic real-time tasks over homogeneous multiprocessors.* Journal of PARC, pages 30–35, 2005. Citeseer.

[96] W. Yuan and K. Nahrstedt. *Energy-efficient soft real-time CPU scheduling for mobile multimedia systems.* ACM SIGOPS Operating Systems Review, vol.37, no.5, pages 149–163, 2003.

[97] B. Schroeder and M.H. Balter. *Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness Cluster Computing*, vol.7, no.2, pages 151–161, 2004. Springer.

[98] W. Tang, Z. Lan, N. Desai and D. Buettner. *Fault-aware, utility-based job scheduling on blue gene/p systems*. IEEE International Conference on Cluster Computing and Workshops, pages 1– 10, 2009.

[99] C. Wang, Z. Zhang, X. Ma, S.S. Vazhkudai and F. Mueller. *Improving the availability of supercomputer job input data using temporal replication*. Computer Science-Research and Development, vol.23, no.3–4, pages 149–157, 2009. Springer.

[100] Supercomputer ’Titans’ Face Huge Energy Costs. http://www.livescience.com/18072-rise-titans exascale- supercomputers-leap-power-hurdle.htm, Accessed: 2012-01-23.

[101] Reducing Energy Consumption and Cost in the Data Center. http://www.datacenterknowledge.com/archives/2014/12/11/reducing-energy-consumption- costdata- center, Accessed: 2014-12-11.

[102] America’s Data Centers Consuming and Wasting Growing Amounts of Energy. http://www.cs.huji.ac.il/labs/parallel/workload/lsdscsp2/index.html, Accessed: 2015-02-06.

[103] The San Diego Supercomputer Center (SDSC) SP2 log. http://www.nrdc.org/energy/data-center efficiency- assessment.asp, Accessed: 2015-02-06.

[104] Logs of Real Parallel Workloads from Production Systems. http://www.cs.huji.ac.il/labs/parallel/workload/logs.html, Accessed: 2015-02-06.

[105] The Intel Netbatch logs. http://www.cs.huji.ac.il/labs/parallel/workload/lintelnetbatch/index.html, Accessed: 2015-02-06.

[106] The ANL Intrepid log. http://www.cs.huji.ac.il/labs/parallel/workload/lanlint/index.html, Accessed:2015-02-06.

[107] Tang, Wei and Desai, Narayan and Buettner, Daniel and Lan, Zhiling. *Job scheduling with adjusted runtime estimates on production supercomputers*. Journal of Parallel and Distributed Computing, vol.73, vol.7, pages 926–938, 2013. Elsevier.

[108] D. Jackson, Q. Snell and M. Clement. *Core algorithms of the Maui scheduler*. Job Scheduling Strategies for Parallel Processing, pages 87–102, 2001. Springer.

[109] L. X. Wang. A course in fuzzy systems. 1999. Prentice-Hall press, USA.

[110] S. Tong, C. Li and Y. Li. *Fuzzy adaptive observer backstepping control for MIMO nonlinear systems.* Fuzzy Sets and Systems, vol.160, no.19, pages 2755–2775, 2009. Elsevier.

[111] S. Tong and Y. Li. Observer-*based fuzzy adaptive control for strict-feedback nonlinear systems*. Fuzzy Sets and Systems, vol.160, no.12, pages 1749–1764, 2009. Elsevier.

[112] L. A. Zadeh. *Is there a need for fuzzy logic?* Information sciences, vol.178, no.13, pages 2751– 2779, 2008. Elsevier.