M.Sc. Engg. Thesis

MANY-OBJECTIVE PERFORMANCE OPTIMIZATION IN HETEROGENEOUS COMPUTING CLUSTERS

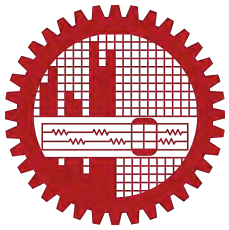
by

A.S.M Rizvi (1015052053 F)

Submitted to

Department of Computer Science & Engineering

(In partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering)



Department of Computer Science & Engineering

Bangladesh University of Engineering & Technology (BUET)

Dhaka 1000

July 22, 2017

Dedicated to my loving parents

Author's Contact

A.S.M Rizvi

Email: asm.rizvi.bd@gmail.com

The thesis titled "MANY-OBJECTIVE PERFORMANCE OPTIMIZATION IN HETEROGENEOUS COMPUTING CLUSTERS", submitted by A.S.M Rizvi, Roll No. **1015052053 F**, Session October 2015, to the Department of Computer Science & Engineering, Bangladesh University of Engineering & Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science & Engineering and approved as to its style and contents. Examination held on July 22, 2017.

Board of Examiners

1. _____

Dr. A. B. M. Alim Al Islam — Chairman (Supervisor)
Associate Professor
Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology, Dhaka.

2. _____

Prof. Dr. M. Sohel Rahman — Member (Ex-Officio)
Head and Professor
Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology, Dhaka.

3. _____

Prof. Dr. Md. Monirul Islam — Member
Professor
Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology, Dhaka.

3. _____

Dr. Muhammad Abdullah Adnan — Member
Assistant Professor
Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology, Dhaka.

4. _____

Prof. Dr. Mohammad Faisal — Member (External)
Professor
Department of Electrical and Electronic & Engineering
Bangladesh University of Engineering & Technology, Dhaka.

Candidate's Declaration

This is hereby declared that the work titled "MANY-OBJECTIVE PERFORMANCE OPTIMIZATION IN HETEROGENEOUS COMPUTING CLUSTERS", is the outcome of research carried out by me under the supervision of Dr. A. B. M. Alim Al Islam, in the Department of Computer Science & Engineering, Bangladesh University of Engineering & Technology, Dhaka 1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

A.S.M Rizvi

Candidate

Acknowledgment

First of all, I would like to express my heart-felt gratitude to my supervisor, Dr. A. B. M. Alim Al Islam, for his constant supervision of this work. He helped me a lot in shaping, deciding steps of my work, and providing infrastructural supports.

I would also want to thank the honorable members of my thesis committee: Prof. Dr. M. Sohel Rahman, Prof. Dr. Md. Monirul Islam, Dr. Muhammad Abdullah Adnan, and specially the external member Prof. Dr. Mohammad Faisal, for their encouragements, insightful comments, and valuable suggestions.

I am also thankful to Mohammad Mosiur Rahman Lunar (Ph.D. Candidate, University of Nebraska at Lincoln, USA), Tarik Reza Toha (Under-graduate student, CSE-BUET), and Siddhartha Shankar Das (Lecturer, CSE-BUET). I sought help from them a number of occasions regarding simulation set-up and performance evaluation of this thesis. I am also grateful to all honorable teachers of the department for their comments and suggestions. I would like to give special thanks to Dr. Muhammad Abdullah Adnan and Abu Wasif for their valuable suggestions during my thesis work.

Last but not the least, I remain ever grateful to my beloved parents, wife and family, for their inspirations behind every success of mine.

Abstract

In computing clusters, there are different performance metrics, which often appear to be conflicting while being attempted to be optimized. For having such conflicting cases along with experiencing existence of heterogeneous environment, it is often difficult for the cluster administrators to select the right number and right combination of machines. As a remedy to this situation, in this thesis, we develop a technique through which cluster administrators can select the right set of machines to enhance cluster performance. In our solution, we integrate both cooling energy consumption and empirical performance characterization of clusters. To the best of our knowledge, existing studies do not integrate these two simultaneously in solving many-objective optimization problem for clusters. We exploit a many-objective optimization approach based on NSGA-III algorithm to solve our cluster problem. Our technique attempts to simultaneously optimize many objectives such as computation time, computation energy, cooling energy, and utilization. Subsequently, we demonstrate through both real experimentation and simulation that our technique mostly performs better than optimization approaches existing in the literature. In this study, we integrate cooling energy while evaluating cluster performance. Cooling energy consumption is one of the most significant parts of total energy consumed by clusters and similar distributed systems. However, little effort has been spent so far to integrate the cooling energy in simulators that are used for simulating the distributed systems. Therefore, we also perform integration of cooling energy consumption in a widely-known simulator of distributed systems namely *SimGrid*.

Acronyms List

ACO = Ant Colony Optimization

BBO = Bio-geography Based Optimization

CT = Current Transformer

EA = Evolutionary Algorithm

KWh = KiloWatt hour

mins = Minutes

MOEA = Multi-Objective Evolutionary Algorithm

MOEA/D = Multi-Objective Evolutionary Algorithm based on Decomposition

NSGA = Non-dominated Sorting Genetic Algorithm

PSO = Particle Swarm Optimization

PT = Potential Transformer

VM = Virtual Machine

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computing clusters are extensively used for distributed and parallel computing now-a-days [3]. In a computing cluster, machines are connected to work together so that they can be viewed as a single machine. Computing clusters are generally used for increasing computation speed, availability, fault tolerance, and scalability. They have a wide range of applicability in simulation, modeling, and experimentation. Examples of applications include galaxy formation simulation, modeling planetary movement, climate change prediction, traffic jam simulation, plate tectonics movement simulation, and experimentation with weather forecast. Fig. 1.1 shows the usage of parallel computing in different modeling, simulation, and experimentation. In traffic jam simulation, we have huge sample space. We have different types of vehicles, different conditions of roads, different pressure of vehicles at different hours of the day, etc. Combining all these things into the simulation is computationally expensive. Using just one machine to simulate all these things is infeasible as well. That is why parallel computing has become necessary and significant in modeling, simulation, and experimentation.

1.1 Background

Parallel computing is necessary for modeling, simulation, and experimentation of complex real-world phenomena where rigorous computing power is necessary [1]. There are different architectures for implementing parallel computing. In the next section, we will describe those

(a) Galaxy formation

(b) Planetary movement

(c) Climate change

(d) Traffic simulation

(e) Plate tectonics

(f) Weather forecast

Figure 1.1: Parallel computing usage in modeling, simulation, and experimentation [1]

architectures.

1.1.1 Architecture of Computing Cluster

In a computing cluster, multiple machines work together to increase overall capacity. Fig. 1.2 shows the architecture of messaging among the machines of a computing cluster. Here, a central machine normally controls the other machines. There is a master machine that distributes a big task among several slave machines. The slave machines work and send their results to the master machine. Master machine generally maintains coordination among the slave machines and accumulate all the results. We can use different tools for the purpose of job distribution. Hadoop [4] and Yarn [5] are such kind of tools. A sophisticated mechanism such as MapReduce [6] normally runs to handle these distribution tasks among machines. A cluster administrator normally operates the cluster. There are different performance metrics which a cluster administrator wants to achieve. Next, we see the performance metrics for computing clusters.

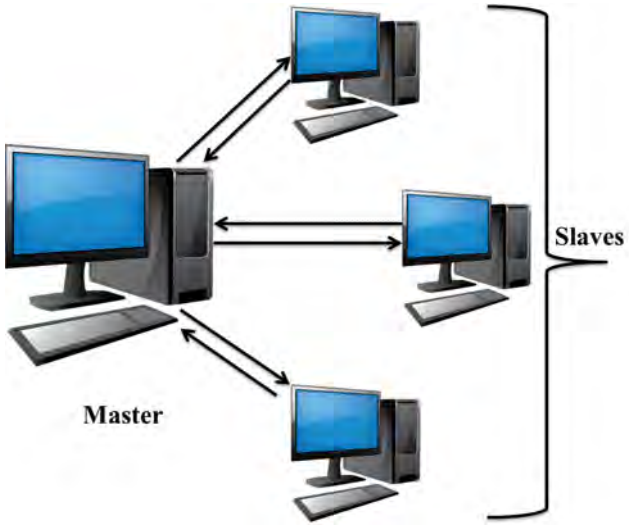


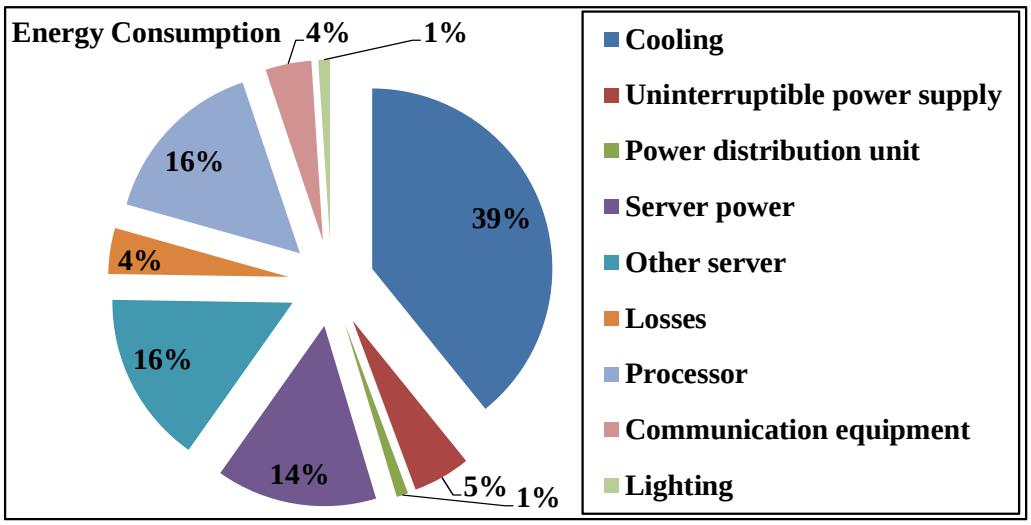
Figure 1.2: Master machine sends tasks to the slave machines. Slave machines will reply after finishing their tasks

1.1.2 Performance Metrics for Computing Clusters

There exist several performance metrics for clusters [7] [8] in the literature. Here, some administrators may want to reduce the total computation energy, few others may want to reduce cooling energy consumption, and so on. In this thesis, we consider both the computation energy and cooling energy in combination. Besides, resource utilization and computation time are also important for some administrators. Therefore, we consider CPU usage and memory usage for resource utilization along with considering computation time.

1.2 Motivation

In many cases, performance objectives of a computing cluster appear to be conflicting. For example, from our year-long collected laboratory data, we find that increasing the number of machines can reduce computation time. It can also decrease total energy consumption. However, increasing the number of machines may decrease resource utilization, and increase cluster maintenance cost as well, pertaining a conflicting scenario with the other objectives. Solving conflicting objectives is a classical problem and there exist myriad techniques in the literature available to solve multiple conflicting objectives [9]. However, little effort has been



(a) Energy consumption in data centers

(b) Data centers from USA consume 39% of their total energy consumption

Figure 1.3: Energy consumption in USA data centers [2]

spent to date to solve conflicting objectives in computing clusters to the best of our knowledge. Therefore, in this thesis, we motivate to propose a new approach to solve the conflicting objectives of a computing cluster.

After getting motivated to give a solution for selecting the right number and right combination of machines, we use our year-long laboratory data to formulate a many-objective optimization problem consisting objectives and constraints relevant to clusters. We perform empirical analyses over the data to facilitate solving the optimization problem. Our solution approach exploits a synergy between greedy method and NSGA-III algorithm. Exploiting the synergy, our solution gives a set of machines as its final output, which cluster administrator can adopt to operate. We test our solutions in a real setup as well as in simulation environment. We find our approach mostly performs better than other existing approaches for computing clusters.

In our thesis, we put a special focus on cooling energy while experimenting with energy consumption. Distributed computing infrastructures such as computing clusters consume a considerable part of electricity consumption around the world and their extent of energy consumption is increasing day by day [10]. For example, an analysis on few data centers reports that around 39% of total energy consumption of the data centers pertains to cooling energy. Fig. 1.3 shows a breakdown of the total energy consumption in US data centers. This energy consumption can vary from place to place, environment to environment, and design to design. Consequently, success of any work related to energy consumption of such infrastructures vastly depends on proper realization of the cooling energy. One of the widely adopted methodologies for realizing different aspects of energy consumption is to perform simulation.

Different simulation tools have been developed for computing energy consumption. Among them NS-2 [11] and NS-3 [12] are the two most popular ones. However, NS-2 and NS-3 are yet to offer any support for measuring energy consumption in wired systems mimicking conventional distributed systems such as computing clusters. Both of these simulators are capable of measuring energy consumption for wireless systems.

SimGrid [13], being a simulation tool developed for simulating distributed systems, offers an energy plug-in for measuring energy consumption in wired systems. However, this energy plug-in is yet to consider cooling energy consumption that remains a significant part of energy consumption as mentioned above. Therefore, in this thesis, we motivate to perform necessary modifications in the plug-in to incorporate cooling energy consumption. Here, we integrate cooling energy consumption with the consideration of environment temperature,

maximum allowable temperature inside the machines, and the workload of the system under consideration. We perform validation of our integration through comparing our simulation results against that of real testbed experiments.

1.3 Objectives of This Thesis

We identify the following objectives for this thesis:

- Propose a solution for the cluster administrator to select the right number and right combination of machines.
- Propose a many-objective optimization problem for cluster computing.
- Put special care for cooling energy which is an objective of the cluster.
- Integrate a cooling energy module in real testbed and simulation environment.
- Propose a solution approach for the many-objective optimization problem so that our solution will perform better.
- Rigorous experimentation to show how our suggested approach performs.

1.4 Our Contributions

Based on our work on devising a new many-objective optimization technique for computing clusters and integrating cooling energy in simulation,

- We formulate a new many-objective optimization problem for computing clusters considering different relevant objectives and constraints. The objectives include cooling energy consumption, which is mostly ignored in contemporary studies. Besides, we consider our year-long collected laboratory data in formulating the problem.
- We develop a new technique exploiting both greedy method and NSGA-III algorithm to solve the many-objective optimization problem for computing clusters. Outcomes

of our technique pinpoint the set of machines that need to be selected to achieve the best-possible performance from a cluster in terms of all the considered cluster objectives.

- We use a well-established simulation platform namely *SimGrid* to experimentally evaluate performances of our proposed and other existing approaches, in diversified settings.

- To perform simulation in *SimGrid* with cooling energy, we point out necessary models that need to be incorporated in *SimGrid* to integrate the cooling energy consumption in it. Subsequently, we make necessary modifications in the existing *SimGrid* to incorporate the models. We simulate different distributed computing systems using our modified *SimGrid* module. Besides, we perform real testbed experiments with similar settings adopted in our simulation. We compare the simulation and testbed results to validate applicability of our proposed modified plug-in of *SimGrid*.

- Finally, we implement our proposed optimization technique along with existing ones in a real setup and evaluate their performances. Comparative analysis over all the experimental results demonstrates that our proposed technique can provide significant performance improvement in most of the cases compared to other existing ones.

The rest of the book is organized in the following way. In Chapter 2, we will show the background and related research studies. After that in Chapter 3, we will discuss about the many-objective problem formulation for the clusters. In Chapter 4, we discuss the methodology that we use to solve the problem which is formulated in Chapter 3. In the later two chapters we will show the experimental results in both real testbed and in simulation environment *SimGrid*. After that we willl have a short conclusion including the future possible research directions.

Chapter 2

Related Work

(a) Large Linux cluster in University of Technology, Germany

(b) Home made cluster

Figure 2.1: Clusters in different size

Distributed computing can be implemented using different platforms such as clusters, grid, and cloud. In clusters, computing machines are connected through a local area network, whereas, in clouds or grids, machines are conventionally geographically distributed [14]. In this thesis, we mainly focus on cluster architectures. We can form clusters not only in a big setup as shown in Fig. 2.1(a) but also in a small home environment as shown in Fig. 2.1(b). We generally build clusters with similar performing machines while it is possible to make grid and cloud with varying performing machines.

As mentioned in Chapter 1, computing clusters as well as distributed systems can have conflicting objectives. In the literature, several research studies exist that intend to solve optimization problem pertinent to these objectives. Here, most of the studies focus on cloud having a little focus on clusters. Next, we present the existing studies.

2.1 Cloud Based Many-Objective Performance Optimization Techniques

Multi-objective optimization techniques are studied for virtual machine based cloud architectures [15]. There exist some other studies [16] [17], [18], which consider resource provisioning techniques in cloud computing. Besides the study presented in [19], focuses on minimizing cost and energy consumption. This study only considers two objectives and is specialized for cloud architectures. In this study, authors try to place the VMs in such a way that it provides the least increase in power consumption without violating the negotiated Service Level Agreements (SLAs). All these studies are not applicable to computing clusters owing to significant architectural gap between clusters and clouds. Moreover, they are not fully aware of cooling energy in cluster computing.

2.2 Cluster Based Many-Objective Performance Optimization Techniques

There are a few studies in the literature, which consider performance optimization in cluster computing. For example, the study in [8] presents a stochastic technique for performance optimization. This study first formulates an energy aware steady-state model. Then, it designs an optimization problem for resource provisioning. The authors introduces an uncertainty model, and they make a stochastic programming formulation based on the steady-state and uncertainty model. Later, an orthogonal weighted sum algorithm was used to generate pareto front solving the optimization problem. This study deals with many objective (more than three objectives [20]) optimization for upgrade cost, failure rate, power consumption, and number of completed tasks objectives. However, it does not consider any specialized many-objective optimization

technique, rather it uses a multi-objective stochastic technique. Nonetheless, existing study in the literature [9], [20] show that a specialized many-objective optimization technique performs better than a multi-objective optimization technique. Existing multi-objective optimization techniques show a number of problems relating to convergence, diversity, and computation time while solving many objective optimization problems [21].

2.3 Cluster and Cloud Based Many-Objective Performance Optimization Techniques

Some recent studies focus on multi-objective performance optimization in computing clusters and clouds. Examples include a Particle Swarm Optimization (PSO) based technique [22] and Ant Colony Optimization (ACO) based technique [23]. In [22], authors solve the problem of load balancing in cloud clusters with the objectives of minimizing the average workload of all servers in cloud clusters, the deviation of the workload, and the migration cost between servers using PSO based technique. In [23], authors simultaneously minimize resource wastage and power consumption in cloud architecture using ACO based technique. However, these techniques are yet to be extended for many-objective cases. Moreover, integrating empirical performance characterization of clusters is yet to be focused by all the techniques in the literature to the best of our knowledge. Such integration of empirical characterization is important as it exhibits a potential to reveal environmental impacts over performance of a cluster.

2.4 Generalized Many-Objective Optimization Techniques

There exist several generalized many-objective optimization techniques in the literature. Examples include MOGA [24], NSGA [25], NPGA [26], etc. Although these methods can be used to find multiple non-dominated solutions on many test problem cases, researchers realized the need of introducing more useful procedure to solve multi-objective optimization problems better. Accordingly, in the study presented in [27], researchers show the notion of elitism helps in achieving better convergence. Later, researchers found many real life problems have more

than three objectives. Optimization problems with more than three objectives are known as many-objective optimization problem [20]. Studies focusing on multi-objective optimization are not sufficient to solve many-objective optimization problems.

Diversity and convergence preservation are two important aspects in solving optimization problem. Research study [28] clearly shows that these two goals are contradictory, and usual genetic operators are not able to attain both goals at the same time. This problem is more severe for many-objective optimization problems. To solve many-objective optimization problems, a research study [29] suggests measures for 5-50 objectives. Other studies such as the study presented in [30] extends NSGA-II [31] using modified diversity-controlling operators to solve 6 to 20-objective problems. NSGA-II possesses some limitations in solving many-objective optimization. Moreover, the study [32] claimed that NSGA-II is not suitable for many-objective optimization problems and suggests a collection of alternative metrics that can replace NSGA-IIs crowding distance operator for better performance. All these techniques are the modifications of the previously-suggested evolutionary algorithms to solve multi-objective optimization techniques. However, these solutions only address special test problems such as DTLZ problems [33]. More real-life and challenging problems are yet to be explored by these techniques. An alternative namely MOEA/D [34] attempts to solve the problems of previous studies, however, it is not tested for a large number of objectives. Finally, a recent technique namely NSGA-III [9] shows better performance for large number of objectives. At the same time, it exhibits capability to work with diverse real-life challenging problems. However, to the best of our knowledge, NSGA-III is yet to be investigated for many-objective optimization in clusters.

As one of the important objectives in this regard is energy consumption, we present an overview on existing studies on simulating energy consumption next.

2.5 Simulation of Energy Consumption in Distributed Systems

Measuring energy consumption has been investigated in various research studies [35], [36], and [37]. However, these studies mostly deals with ad-hoc networks. For example, the study pre-

sented in [38] presents an energy consumption model for mobile ad-hoc networks for measuring performance of routing protocols. This study uses NS-2 simulator for measuring energy consumption. Similar other study [39] also exists in the literature. Besides, some other studies focus on energy efficient protocols in wireless networks [40]. Nonetheless, only a few studies [41] aim at modeling energy consumption in distributed systems. An example of such studies [42] develops different APIs that are used in *SimGrid*. However, to the best of our knowledge, none of these studies has modeled the cooling energy to integrate it within total energy consumption. Nonetheless, *SimGrid* energy module has already been used by the studies presented in [43], [44], hence, integrating cooling energy with *SimGrid* is necessary to get more concrete and accurate results.

Chapter 3

Problem Formulation

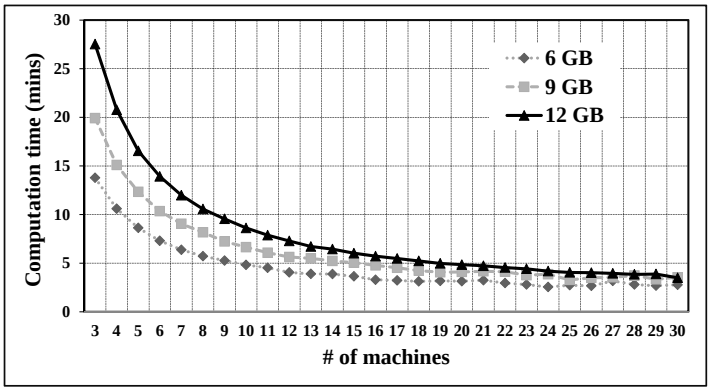
To formulate our research problem, we first conduct laboratory experiments for more than a year. In our experiments, we measure different metrics such as computation time, energy consumption, and resource utilization. We utilize the measured data to identify impacts of operational parameters on objectives of the cluster. Our experimental data demonstrate that objectives of a cluster get substantially influenced by environmental impacts. Here, we consider the number of machines, configuration of machines, network bandwidth, etc, as the operational parameters.

As outcome of our study vastly depends on definition of objectives of the cluster, we describe the objectives that we consider in our cluster. Then, we construct a mathematical model incorporating these cluster objectives, which eventually formulate our research problem. Note that in our model and problem formulation, more cluster objectives can be added if needed. However, in this thesis we confine our focus with four cluster objectives.

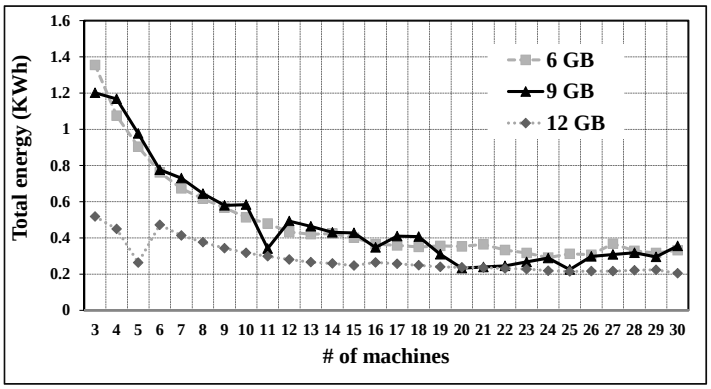
3.1 Cluster Objectives

We present the four cluster objectives which we adopt in this study, along with impacts of operational parameters over them, in the following way:

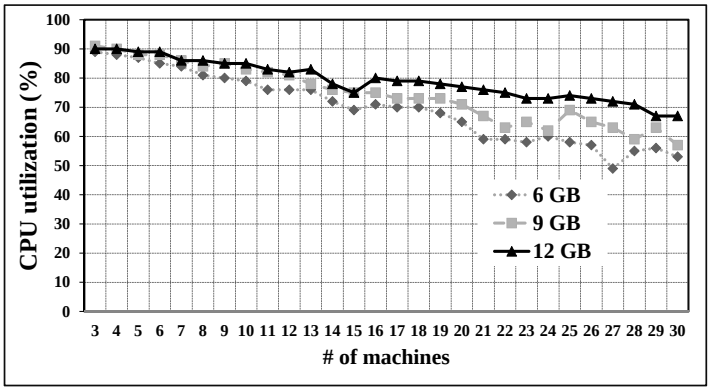
- Objective 1 - Decreasing computation time: Computation time refers to the time required to finish a task assigned to a cluster. Cluster administrators always want to decrease the



(a) Computation time comparison



(b) Total energy consumption comparison



(c) CPU utilization comparison

Figure 3.1: Comparative analysis of cluster objectives with an increase of number of machines

computation time so that tasks are finished within a shortest possible time. From Fig. 3.1(a), we can see that when we increase the number of machines, the computation time gets decreased. We make our experiment in different seasons of the year and find the same pattern which indicates the robustness of our experimental decision. Moreover, if we increase the workload, the computation time gets increased as well.

- Objective 2 - Decreasing total energy: In computing clusters, we want to decrease total energy consumption. From Fig. 1.3, we can see cooling energy plays a significant role in total energy consumption (around 39%). Hence, we incorporate cooling energy module in total energy consumption. We describe the procedure to integrate cooling energy into the existing *SimGrid* in Chapter 5. From Fig. 3.1(b), we can see a trend line of decreasing energy consumption with the increase of number of machines. Decreasing energy consumption with the increase of number of machines is unusual, however, it was evident in our experiment. We describe a possible cause in Chapter 5.

- Objective 3 - Decreasing cost: Cost to operate a computing cluster increases with the increase of number of machines. Cluster administrators want to decrease the cost of a cluster. With the increase of number of machines other costs like maintenance, utilities etc. will also be increased.

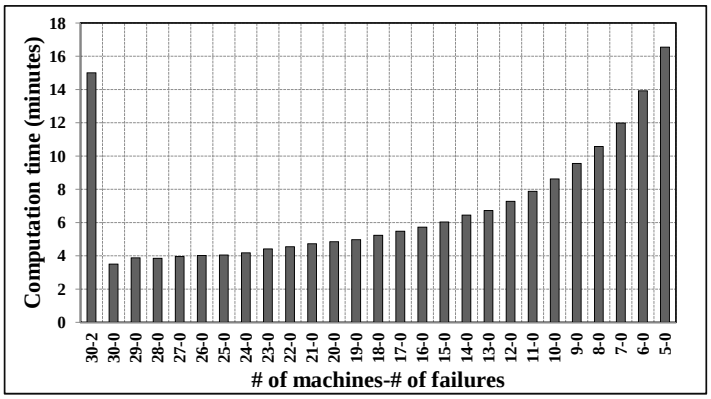
- Objective 4 - Increasing utilization: Administrators also try to improve the resource utilization. When a machine runs it consumes a particular amount of physical memory and CPU usage. Getting more CPU and memory utilization with the help of fewer number of machines is more desirable than getting smaller CPU and memory usage with a greater number of machines. From Fig. 3.1(c), we can see CPU utilization decreases with the increase of number of machines.

In the following section we will discuss different environmental impacts over cluster objectives.

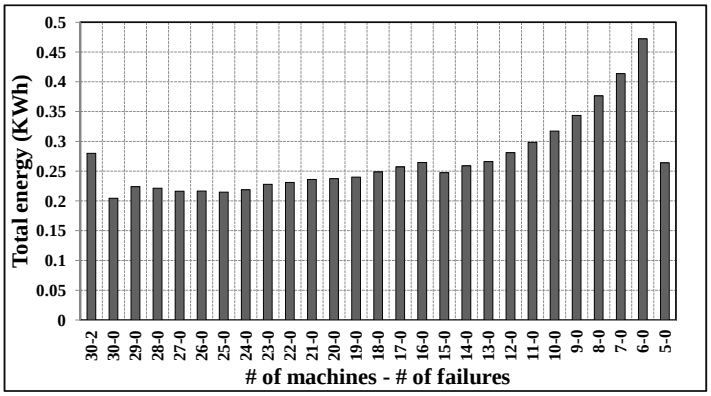
3.2 Impacts of Different Factors on Cluster Objectives

Different environmental factors have significant impacts over cluster objectives. Here, we discuss how these factors have impacts on cluster objectives.

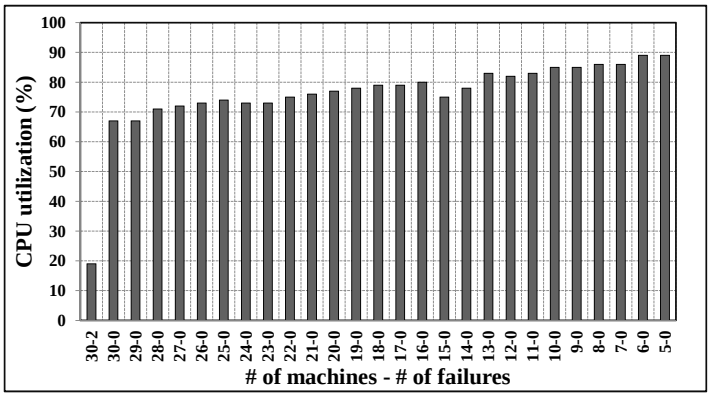
Impacts of Number of Machines: Cluster objectives are dependent on the number of operating machines. Number of operating machines has conflicting impacts over cluster objectives. By increasing number of machines, we can decrease computation time. At the same time, the



(a) Computation time comparison



(b) Total energy consumption comparison



(c) CPU utilization comparison

Figure 3.2: Comparative analysis of cluster objectives with different number of machines and with a presence of machine failure

total energy consumption will be decreased. We can fulfill these two objectives with the increase of number of machines. However, resource utilization will be decreased with the increase of number of machines. As more machines are now working, they will not be utilized according to their capacities. At the same time, the operating cost will also be increased, which is not

desirable. Hence, number of machines have conflicting impacts over cluster objectives. Fig. 3.1 shows the impacts of increasing number of machines.

Impacts of Configuration of Computing Machines: Configuration of computing machines have impacts on cluster objectives. High-performing computing machines can reduce the computation time. At the same time, these machines increase the overall cluster cost.

Impacts of Machine Failures: From our experiment, we see that failures can have a serious impact on computation time, total energy consumption and also in resource utilization. However, these impacts are highly dependent on workload. We find that impacts with smaller workload are more severe than the impacts with larger workload. From Fig. 3.2(a) we can see a comparative analysis. With workload of 12.6 GB, 30 machines with 2 failures have comparable performance to 6 machines with no failure. Here, along the X-axis, M-N denotes M number of machines with N number of failures. Hence, 30-2 denotes 30 machines with 2 failures. From Fig. 3.2(b), we can see that failures can cause a significant increase in energy consumption. Even 12 machines with no failure can be more energy efficient than 30 machines with 2 failures. From Fig. 3.2(c), we can also see the impact of failures in CPU utilization.

Based on these impacts we develop a mathematical model for many-objective optimization problem.

3.3 Proposed Problem Formulation

We define the configuration of a computing machine as the machine property. We can have different machine properties such as physical memory, processor speed, failure rate or network bandwidth. We represent these properties as $P_1, P_2, P_3, ..., P_N$, where N is the number of machine properties. We express the effectiveness of a computing machine using the term Machine Value, MV. We can define the machine value of a computing machine i by $MV_i = w_1 \times P_1 + w_2 \times P_2 + w_3 \times P_3 + ... + w_N \times P_N$. $w_1, w_2, w_3, ..., w_N$ are the weights of N properties. In this thesis, we use CPU usage, memory usage, and network bandwidth as machine properties. We try to optimize the following four objectives in this thesis: (1) Minimizing computation time, (2) Minimizing total energy consumption, (3) Minimizing cost, (4) Increase utilization.

Minimizing computation time: Computation time is dependent on the number of machines and machine value. As we said earlier, we will get a decrease in computation time with the

increase of number of machines. Computation time will also be decreased when the machine value is increased (using high performing machines). Let, there are N_M machines in the cluster. We can express the objective as:

$$\max \quad \sum_{i=1}^{N_M} s_i(MV)_i \tag{3.1}$$

s_i is the decision variable, which describes an indicator function. s_i is 1 if we take i^{th} machine into our cluster and 0 if we do not take i^{th} machine. We can write:

$$s_i = \begin{cases} 1, & \text{if machine } i \text{ is in the cluster} \\ 0, & \text{otherwise} \end{cases}$$

For simplicity, we make this optimization problem as a minimization problem and make necessary changes in all the objective functions. Hence, the objective function becomes:

$$min \quad \sum_{i=1}^{N_M} (100 - s_i(MV)_i) \tag{3.2}$$

The machine value for any device i will be any value from 0 to 100. Computation time is also dependent on work load. If the workload gets increased then the computation time will be decreased. If the workload for machine i is W_i, then we get the following normalized value for the objective function:

$$\min \quad \frac{\sum_{i=1}^{N_M} (100 - s_i(MV)_i)}{\sum_{i=1}^{N_M} s_i \times 100} \times PW_1 + \frac{W_i}{W_{max}(i)} \times PW_2 \tag{3.3}$$

We use $W_{max(i)}$ as the maximum allowable workload for machine i. W_i is the machine workload, which we assume $\frac{Total_{workload}}{\sum_{i=1}^{N_M} s_i}$. This is because our simulation platform *SimGrid* normally distributes its workload among machines equally. PW_1 and PW_2 indicate the property weights. We assume similar effects from machine configuration and workload. That is why we give same 50-50 weights in both PW_1 and PW_2 of Eq. 3.3. We can also use a variable here if we intend to make different effects of machine configuration and workload.

Minimizing total energy consumption: Energy consumption is dependent on the number

Table 3.1: Impacts of weight values in experimental results.

Weight ratio	**Compu tation time**	**Cooling energy (KWh)**	**Computation energy (KWh)**	**No. of selected machines**	**CPU usage (%)**	**Memory usage (%)**
25-75	34.8	2.6	0.3	7	57.5	91.2
75-25	23.5	1.76	0.24	11	55.2	91.0

of cluster machines and temperature difference. From Fig. 5.4(b), we can see that total energy is decreasing with the increase in number of machines. If the temperature difference (T_{diff}) decreases, the total energy consumption gets decreased. Let, T_E be the environment temperature. In our model, there is a range for allowable temperature. T_M is the maximum allowable temperature within the cluster, and T_L is the lowest value of allowable temperature range. T_D be the decision variable within T_L and T_M. We can describe $T_{diff} = T_E - T_D$. The cooling system needs to cool the system by T_{diff} amount. The maximum temperature difference can be $T_{MaxDiff} = T_E - T_L$. We can write the following objective function:

$$\begin{aligned} \min \quad & \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times PW_3 + \frac{T_{diff}}{T_{MaxDiff}} \times PW_4 \\ \text{subject to} \quad & T_L \leq T_D \leq T_M \end{aligned} \tag{3.4}$$

We assume same weights for number of machines and temperature difference (PW_3 and PW_4) in Eq. 3.4. We can use different weights if we intend different effects of number of machines and temperature difference.

We can bias the solution for a specific objective by varying the weights. In Table 3.1, we can see the impacts of weight values in experimental results. 25-75 weight ratio indicates to give less weights (25%) in variables PW_1 and PW_3. It gives around 75% weight to variable PW_2 and PW_4. This 25-75 weight ratio selects less number of machines. Hence, computation time, and energy consumption gets increased as we select less number of machines. At the same time, CPU usage, and memory usage gets increased as we select less number of machines. 75-25 weight ratio give the opposite results.

Minimizing cost: Cost of the cluster is dependent on the number of machines and machine configuration. Cost will be increased with the increase of number of machines. High performing

machines will also increase the cost. We can write the objective function:

$$\min \quad \sum_{i=1}^{N_M} s_i(MV)_i \tag{3.5}$$

Maximizing utilization: Utilization will be increased if we decrease the number of machines. We multiply with 100 here to have a value within 0 to 100 range as we have all other objective values within this range. $\sum_{i=1}^{N_M} s_i$ indicates the number of selected machines and N_M indicates the number of total cluster machines. We can write the objective function as:

$$\min \quad \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times 100 \tag{3.6}$$

If there are $\sum_{i=1}^{N_M} s_i$ number of machines and W_i workload, then there should be $\frac{W_i}{\sum_{i=1}^{N_M} s_i}$ work load per machine approximately. If HDD_i be the size of the hard disk of machine i , we can write the constraint as $\forall i \frac{W_i}{\sum_{i=1}^{N_M} s_i} \leq HDD_i$. Combining all the above objectives and constraints we can write:

$$min \begin{cases} \frac{\sum_{i=1}^{N_M} (100 - s_i(MV)_i)}{\sum_{i=1}^{N_M} s_i \times 100} \times PW_1 + \frac{W_i}{W_{max}(i)} \times PW_2 \\ \\ \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times PW_3 + \frac{T_{diff}}{T_{MaxDiff}} \times PW_4 \\ \\ \sum_{i=1}^{N_M} s_i(MV)_i \\ \\ \frac{\sum_{i=1}^{N_M} s_i}{N_M} \times 100 \end{cases}$$

$$subject\ to \begin{cases} \forall i \frac{W_i}{\sum_{i=1}^{N_M} s_i} \leq HDD_i \\ \\ \sum_{i=1}^{N_M} s_i(MV)_i > 0 \\ \\ T_L \leq T_D \leq T_M \end{cases}$$

In Chapter 4, we will try to optimize the above optimization problem.

Chapter 4

Proposed Solution Technique

In Chapter 3, we described a many-objective optimization problem and there are many techniques to solve this problem which can be found in [20]. In our experiment, we use NSGA-III as our baseline algorithm. We make empirical analyses of the cluster environment to modify the NSGA-III algorithm. We use a greedy approach while modifying the existing NSGA-III algorithm.

Justification behind our Proposed Approach: We use a synergy between NSGA-III and greedy approach in solving our cluster problem. Applying greedy approach in evolutionary algorithm is not new. In [45], authors use a greedy crossover approach to solve traveling salesman problem [46]. In [47], the authors try to use a greedy approach with evolutionary algorithm in solving bounded-diameter minimum spanning tree problem. Authors try to solve a quadratic assignment problem based on greedy genetic algorithm in [48]. In most of the cases, later in this book in Section 6.1, we also show our greedy NSGA-III approach performs better than the existing NSGA-III algorithm which also validates our solution approach.

Several functions are usually participated in an optimization algorithm like selection, crossover and mutation. We use our analyses results to design these functions. In our optimization problem, selection decision of a machine is considered as a decision variable. If there are N_M number of machines then we will have N_M number of decision variables. We also consider the expected environment temperature (previously expressed as T_D) as a decision variable. This expected temperature will be maintained by the cooling devices. Hence, our algorithm will finally give the number of machines to be active in the cluster, a selected set of machines and a

temperature, which needs to be maintained by the cooling devices. We will have the following decision variables:

$$s_1, s_2, s_3, ..., s_{N_M}, T_{AC}$$

Here, $s_1, s_2, s_3, ..., s_{N_M}$ are the indicate variables which can be either 0 or 1. T_{AC} indicates the temperature which should be kept by the cooling devices. Now, we are describing different modifications over the functions which are actively used in optimization steps.

Population Selection From our experiment, we find that if the number of machines is below than a particular number then the computation time and total energy consumption are very high. From Fig. 3.1(a) and 3.1(b), we can see that the rate of decrement in computation time and total energy is very high when the number of machines is less than 6 to 8. After this range, we see a decrease in the changing rate significantly. Based on this observation, while we select population for the next generation, we eliminate the population which has fewer operating machines. In our case, we take this threshold value, T_h as $\frac{N_M}{6}$. We use jMetal [49] as the

Algorithm 1 Population Selection

```
1: function selectPopulation(Threshold value, T_h)
2:     G ← Existing generation
3:     for each population p ∈ G do
4:         populationSize ← populationSize(p)
5:         if populationSize > T_h then
6:             newGeneration.add(p)
7:     Fill up the generation with random population
8:     return newGeneration
```

objective optimization framework and modify the existing selection, mutation and crossover functions. Algorithm 1 shows the steps for population selection.

For a workload of 67.7 GB, we experiment in the simulation platform *SimGrid* to show the effects of different threshold values. We will present the details of experimental setup in later chapters. For now, we are just going to show the simulation results for threshold values of N_M from 2 to 8.

We can see the impacts of selection threshold in Table 4.1. We have a good value for computation time when we have N_M as 4. We have good values for cooing energy and computation energy when N_M is 6. When we take very small value for threshold N_M, chance is very high

Table 4.1: Impacts of selection threshold value in experimental results

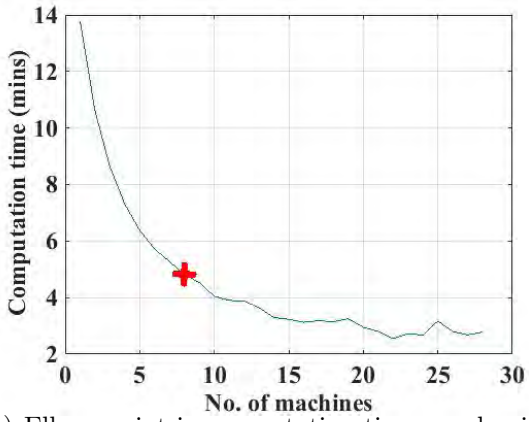
Threshold value	**Computation time (mins)**	**Cooling energy (KWh)**	**Computation energy (KWh)**
2	11.1	0.43	0.92
4	9.5	0.31	0.66
6	9.7	0.24	0.53
8	10.2	0.32	0.68

Table 4.2: Impacts of selection threshold value in solutions of pareto-front

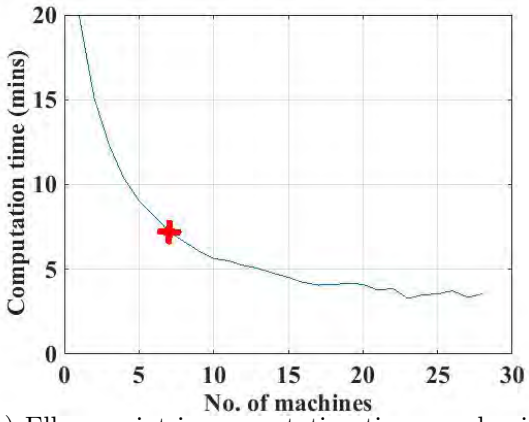
Threshold value	**Total no. of solutions in the pareto front**	**No. of solutions with $<$ 6 machines**	**No. of solutions with $>=$ 6 machines**
2	120	34	86
4	120	27	93
6	120	13	107
8	120	14	106

to have more population with with fewer number of machines. Table 4.2 shows the solutions of pareto-front. It is evident that when N_M is 2, there are 34 solutions among 120 solutions which have less than 6 machines in its pareto-front. When we increase the value of N_M, the number of solutions with less than 6 machines gets decreased. We found the lowest value when we have N_M as 6. Solutions with less than 6 machines have bad effects over cluster objectives. Hence, we try to ignore these solutions as our solution for the cluster administrator. We can use the elbow points to get the value of N_M. Before this elbow point we have a sharp change in the output. After this point, we have a smooth change. Fig. 4.1(a) to 4.1(f) show the values of elbow points. It indicates the elbow points in between 7 and 10.

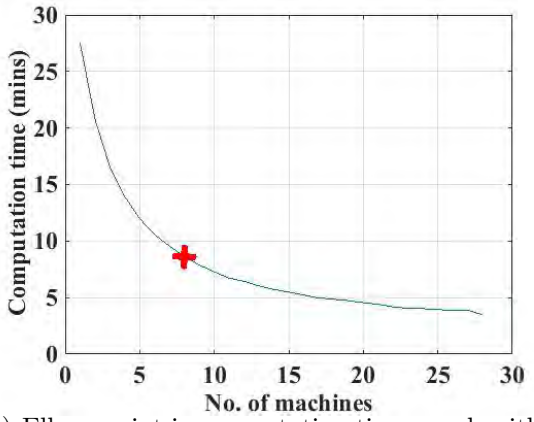
Crossover: While making crossover we use a partition to separate machine selection variables and temperature variable as they are different forms of variables. We make crossover among machine selection variables while not mixing temperature variable with machine selection variable. From Fig. 4.2, we can see the crossover regions. Yellow portion and blue portion will have crossover separately. From Algorithm 2, we can see the crossover algorithm. We make three steps crossover modification. Following a greedy technique, we introduce biasness for the highly performing machines while we make crossover. This ensures a high chance of their



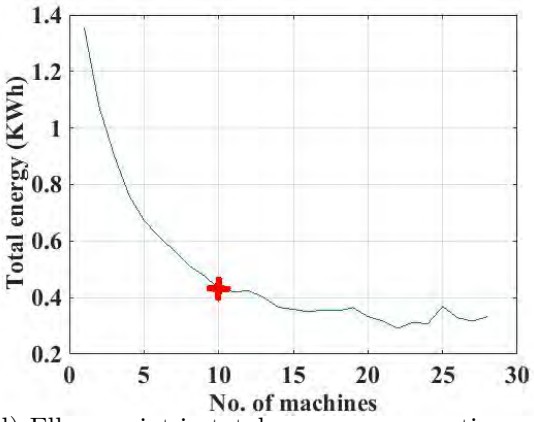
(a) Elbow point in computation time graph with 6 GB workload



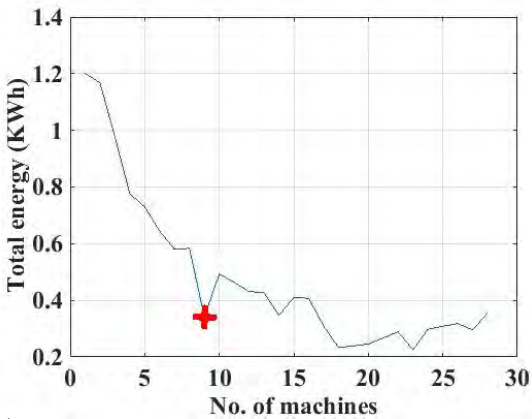
(b) Elbow point in computation time graph with 9 GB workload



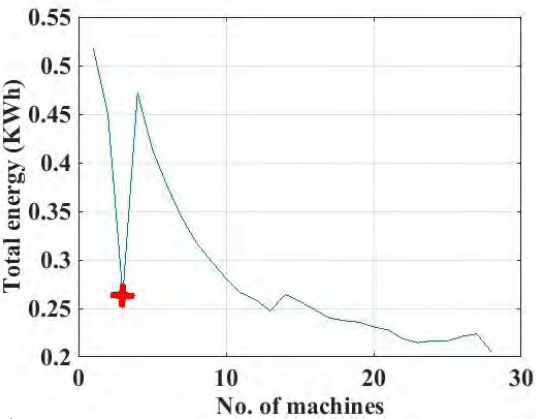
(c) Elbow point in computation time graph with 12 GB workload



(d) Elbow point in total energy consumption graph with 6 GB workload



(e) Elbow point in total energy consumption graph with 9 GB workload



(f) Elbow point in total energy consumption graph with 12 GB workload

Figure 4.1: Elbow points in different graphs for getting the value of N_M

Algorithm 2 Population Crossover

```
1: function crossOver(Parent P_1, Parent P_2)
2:     C_1 ← Chromosome set in P_1
3:     C_2 ← Chromosome set in P_2
4:     size ← C_1.size() − 1
5:     i = 0
6:     while i < size do
7:         if C_1.get(i) ≠ C_2.get(i) and Random.nextDouble() < crossOverProbability then
8:             Inter-change chromosome value
9:         noOfCluster ← 2
10:        Implement k-means clustering for all machine properties with noOfCluster
11:        Find out the machines, which are in the top group for all the machine properties, S_T.
12:        Find out the machines, which are in the lowest group for all the machine properties, S_L.
13:        if i is in S_T and Random.nextDouble() < insertProbability then
14:            C_1(i).set(1)
15:            C_2(i).set(1)
16:        if i is in S_L and Random.nextDouble() < deletionProbability then
17:            C_1(i).set(0)
18:            C_2(i).set(0)
19:        i++
20:    if Random.nextDouble() < takeTopProbability then
21:        rndProperty = Random.nextInt() mod noOfproperty
22:        Take the top machine for rndProperty
23:        Remove the lowest valued machine for rndProperty property
24:    Update P_1 and P_2 according to the value of C_1 and C_2
```

selection in the new generation. We describe the three steps crossover below:

- **Half uniform crossover:** We apply half-uniform crossover as our primary crossover technique. Crossover is made over 50% of the total chromosomes. As we have binary decision variable, we only make crossover when the particular chromosome is different from each other. We can see the half uniform crossover in lines 7 and 8 of Algorithm 2.

- **Crossover based on clustering:** We separate the N_M number of machines into two clusters for each property. Line 10 of Algorithm 2 shows this. We use Euclidean distance in k-means clustering to make two separate clusters. We deploy Cluster 3.0 [50] tool for clustering. We name these two clusters as high performing and low performing clusters. We make a set (S_T) of machines, which are in the high performing cluster for all properties

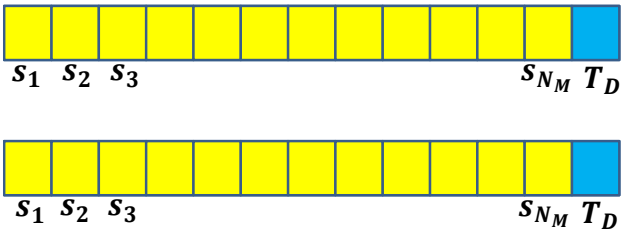


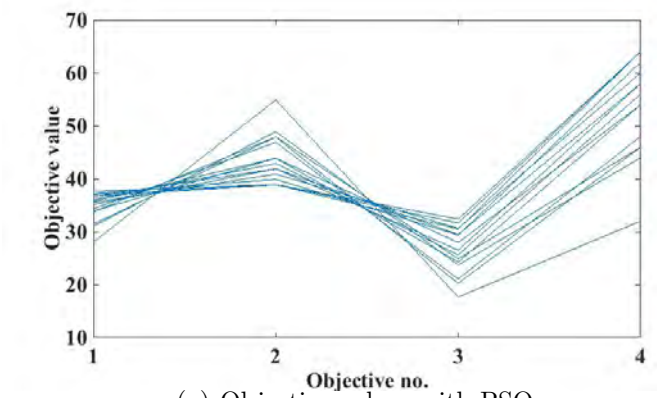
Figure 4.2: Crossover region for decision variables. Crossover will be made within yellow portion and blue portion separately.

and a set of machines (S_L), which are in the low performing cluster for all properties. Line 11 and 12 of Algorithm 2 show this. From S_T, with some probabilistic condition, we take machines into our next generation. Besides, from S_L, with some probabilistic condition, we do not take that machine into our next generation. We can see this in lines 13-18 of Algorithm 2.

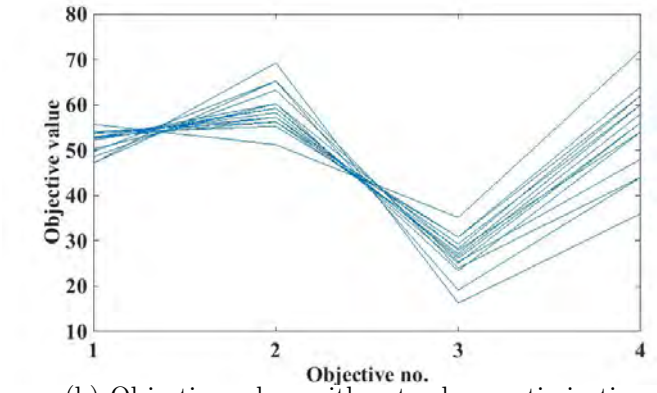
- **Take the top and remove the bottom:** With some probabilistic condition, we take the topmost machine for one property (which is also selected with some probabilistic process) and exclude the bottom most performing machine for that property. We see this from line number 20-23 of Algorithm 2.

Solution Filtering: Optimization methods give a pareto-front with multiple solutions. From Fig. 4.3, we can see four objective values for different optimization techniques. In these graphs, each line indicates one solution with four objective values. We only show fifteen solutions with all the objective values for the sake of clarity. Along the X-axis we take the objective numbers, and along the Y-axis we take the objective values. Solutions having a desirable value for one objective sometimes have an undesirable value for other objectives. These solutions are not acceptable since they make significant performance degradation for some objectives. Hence, we take only those solutions, which have values from 25% to 75% for all objectives. Among these solutions, based on the administrator defined weighted function, we select only one solution. We use the following function to converge four objective values into one value:

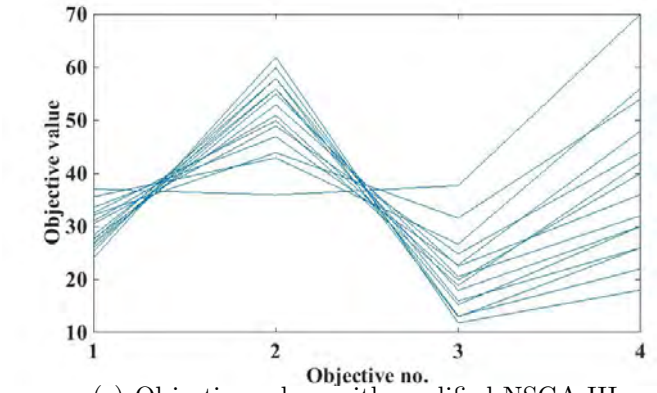
$$F_{total} = W_{obj1} \times V_{obj1} + W_{obj2} \times V_{obj2} + \cdots + W_{objN} \times V_{objN}. \tag{4.1}$$



(a) Objective values with PSO



(b) Objective values with ant colony optimization



(c) Objective values with modified NSGA-III

Figure 4.3: Objective values with different optimization techniques

In Eq. 4.1, every objective value, V_{obj}, should be within 25% to 75% of the corresponding objective value. In this way, we can filter out the solutions which do not have desirable values for all objectives. The weights of these objectives can be defined by the cluster administrator. After having the feasible solutions, we make a sorting over the $feasibleSolution$, and take the top solution having the lowest merged objective value since we have a minimization problem

here. Algorithm 3 shows the filtering process. Here, $firstBoxPlot$ indicates the 25% value and $thirdBoxPlot$ indicates the 75% value.

Algorithm 3 Solution Filtering

```
1: function SOLUTIONFILTERING(objectiveValues, O)
2:     if for all objectives o.val() ≥ O.firstBoxPlot() and o.val() ≤ O.thirdBoxPlot() then
3:         feasibleSolution.add(O)
4:     Sort feasibleSolution in increasing order
5:     return feasibleSolution.get(0)
```

Chapter 5

Cooling Energy Integration in *SimGrid*

In the previous chapters, we mentioned that we integrate a cooling energy module into *SimGrid* to evaluate total energy consumption in clusters correctly. In this chapter, first, we will give an overview about *SimGrid*. Then we will show how we integrate cooling energy module into *SimGrid*. Later, we will show how we validate our model with real testbed setup.

5.1 Overview on *SimGrid*

SimGrid [13] is a well-known simulation tool for distributed systems. It mainly supports C and Java programming language. Users can create simulation topology using XML format. To create topology, one must define the configurations of all the nodes along with the connections among them. One must describe a deployment scenario of the topology in another XML file. In XML format, one needs to mention about the description of jobs, their chunk sizes, selection of master, and slave nodes. SimGrid master distributes the jobs among the worker machines. *SimGrid* can measure the computation time to complete the jobs. Moreover, there is a module to calculate computation energy consumption. This energy uses workload and computation time while evaluating computation energy consumption. Though *SimGrid* can measure computation energy, it does not have any plug-in to measure cooling energy. Next, we will show the methodology that we use to measure cooling energy of computing clusters.

5.2 Proposed Methodology

As mentioned earlier, *SimGrid* can evaluate energy dissipations by the machines of a distributed system. However, this energy plug-in does not include cooling energy. Present energy plug-in first evaluates the CPU load. CPU load is dependent on the task that is intended to be done by the participant machines. Then the energy plug-in measures the necessary time to complete the task. This time is dependent on the processing power of the machines. Based on the CPU load and required time, *SimGrid* measures the energy consumption to complete the given task.

SimGrid has three types of energy consumption state. One state gets initiated when the CPU is in full-load, another state gets initiated when the CPU is idle, and the remaining state gets initiated when the CPU is totally off. All of these states are pertinent to measuring computational power, exhibiting no impact related to cooling power.

In *SimGrid*, we model cooling energy based on CPU load and temperature difference between environment and cluster temperature. Cooling energy is also dependent on many other factors, however, to correlate with the existing simplified model of energy consumption, we keep our model simple. We have the conventional heat formula as:

$$Q = C_p \times W \times DT \tag{5.1}$$

where Q denotes the generated heat, C_p means the specific heat. W refers the mass of the airflow per minute. This airflow is required to keep the temperature of the machines to a specific value, and the mass of the airflow is counted for a particular time. DT means the difference between environment temperature and maximum allowable temperature (we will give an example later).

Also, it is not very hard to write:

$$W = CFM \times D \tag{5.2}$$

Here, CFM denotes cubic feet per minute and D refers to density. We can validate this equation by observing the units. CFM refers to $volumeperminute$ and density refers to $masspervolume$. Multiplying both we get $massperminute$ which is the same unit that we have for W.

Putting the value of Eq. 5.2 to Eq. 5.1, we get the following equation:

$$Q = C_p \times CFM \times D \times DT \tag{5.3}$$

From Eq. 5.3, we consider the equation for airflow in a chassis as follows:

$$CFM = \frac{Q}{C_p \times D \times DT} \tag{5.4}$$

Specific heat of a room remains constant and at a certain place where the pressure is constant we can assume density also remains constant. Considering other heat losses through the chassis wall we can write the following formula [51] for Fahrenheit scale:

$$CFM = \frac{3.16 \times Q}{DT(F)} \tag{5.5}$$

Where $DT(F)$ denotes the temperature difference in Fahrenheit scale. In Celsius scale it can be written as:

$$CFM = \frac{1.76 \times Q}{DT(C)} \tag{5.6}$$

For example, if the chassis has 200 watts of load, environment has temperature of 26° Celsius, and maximum allowable temperature is 38° Celsius, then in Celsius scale we can write:

$$CFM = \frac{1.76 \times 200}{(38 - 26)} = 44 \tag{5.7}$$

CPU load is responsible for heat generation. More heat will be generated with the increase of CPU load. As they show proportional relation, we can write Q as the CPU load instead of heat, for any given time and task.

Algorithm 4 shows the algorithm that we used in our experiment. Here, environment temperature is T_E and we need to keep the chassis temperature at T_A, which is the maximum allowable temperature. Then the difference will be $T_E \sim T_A$. This difference is the same difference that is showing in Eq. 5.5 and Eq. 5.6 as T_F annd T_C. *SimGrid* provides the CPU load. Using this CPU load and temperature difference we calculate the value of CFM. We use

Algorithm 4 Cooling Power Integration Algorithm

1: **function** UPDATECOOLINGENERGY(Work-load, W)
2: $T_E \leftarrow EnvironmentTemperature$
3: $T_A \leftarrow MaximumAllowableTemperature$
4: $T_{Diff} \leftarrow T_E \sim T_A$
5: $CFM \leftarrow \frac{1.76 \times W}{DT(C)}$
6: $Power \leftarrow PreviousPower + (CFM \times 47.82)$
7: $ConsumedEnergy \leftarrow Power \times TimeForComputation$

Table 5.1: Simulation environment in laboratory

Parameter	**Value**
No. of master	1
No. of slaves	29
Processor	Intel Core 2 Duo
Processor base frequency	2.4 GHz, 2.66 GHz and 2.8 GHz
Memory	1GB to 2GB
OS	Ubuntu 14.04 LTS (x86)

Table 5.2: Frequency and memory vs number of machines

Processor base frequency	**Memory**	**No. of machines**
2.4 GHz	1 GB	1
2.4 GHz	2 GB	2
2.66 GHz	1 GB	2
2.66 GHz	2 GB	8
2.8 GHz	2 GB	17

1 CFM = 47.82 W [52] to get the watt value corresponding to the CFM. Finally, we multiply this watt value with computation time to get the consumed energy.

5.3 Validation of Cooling Energy Integration

We validate our energy integration through comparing our simulation results against that obtained from real experiments performed in a real testbed. We briefly present the testbed settings following a comparative result.

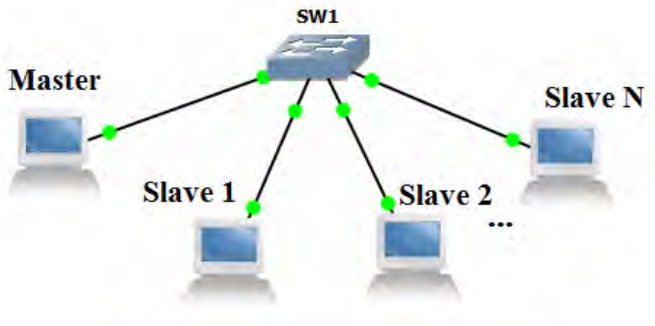


Figure 5.1: Topology of laboratory setup

5.3.1 Testbed Settings

We prepare a testbed in a laboratory having thirty machines. Among them we select one as the master node and rest ones as slave nodes. Fig. 1.2 shows the distribution of tasks from the master node to slave nodes. To implement the distributed system, we use Hadoop [4] framework. We set up a distributed, multi-nodes (30 PCs) Apache Hadoop cluster backed by the Hadoop Distributed File System (HDFS), running on Ubuntu Linux [53]. We vary the data size from 3.14GB to 12.6GB (with 4, 8, 12 and 16 files, each having 787 MB data) and the machine number from 5 to 30. We exploit this laboratory setup to get total energy and cooling energy consumption. For a particular data size and number of machines, we run the popular word-count task four times so that we can minimize the effects of outliers. Table 5.1 shows the lab environment in detail, Table 5.2 shows the number of machines based on frequency and memory, and Fig. 5.2 shows the snapshot of laboratory set-up. From Fig. 5.1, we can see the overall topology where a master and 29 slaves are connected through a switch.

We use two Arduino energy monitors to get computation energy and cooling energy. Each energy monitor has two current transformers and one potential transformer. In Fig. 5.3, left energy monitor collects current from air-conditioner #1 (AC-1), cluster current, and cluster voltage. Right energy monitor collects current from air-conditioner #2 (AC2), air-conditioner #3, and voltage from air-conditioner #2. We use these hardware tools to estimate the values of real power, apparent power, power factor, voltage, and current. We use power formula to get computation power and cooling power. Power consumed by air-conditions is considered as the cooling power. After that, using $Work = P \times T$ formula we get the computation energy and cooling energy, where, P refers to power and T refers to time. One thing is to be noted

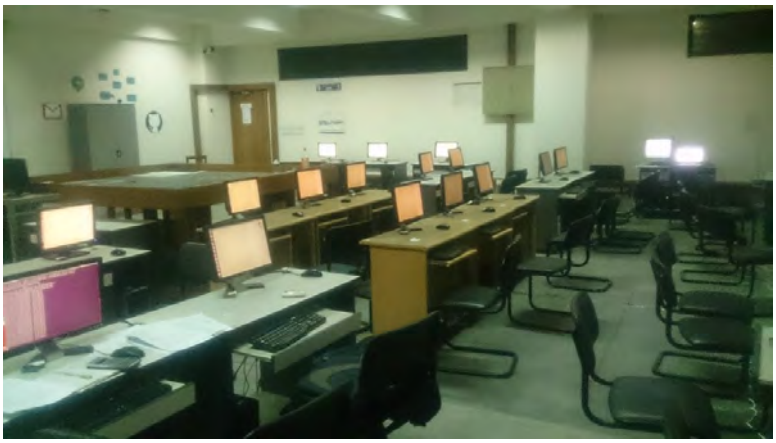


Figure 5.2: Snapshot of laboratory setup

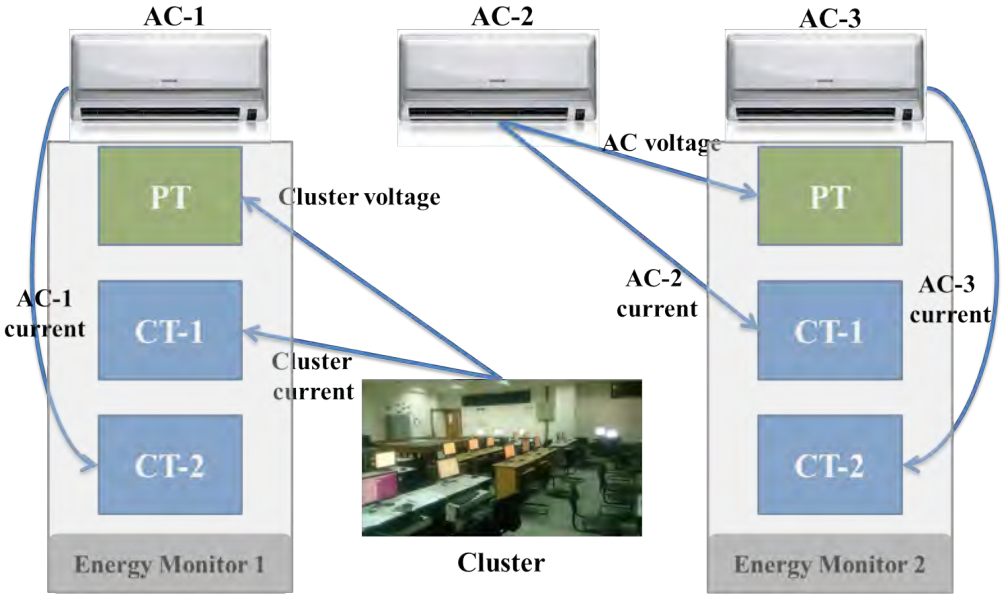
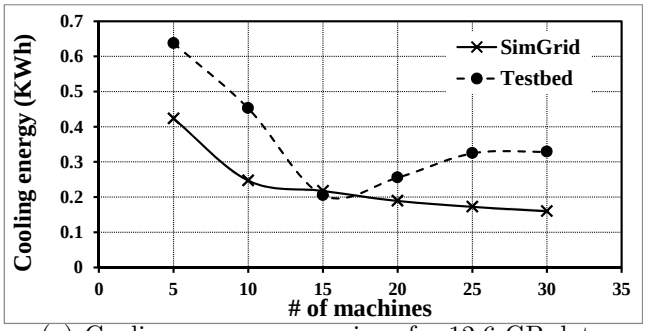
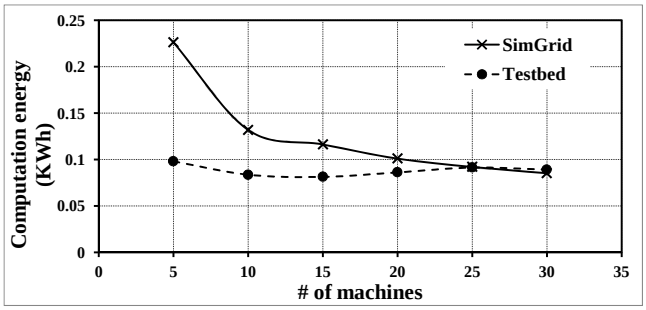


Figure 5.3: Testbed hardware setup for evaluating cooling energy and computation energy

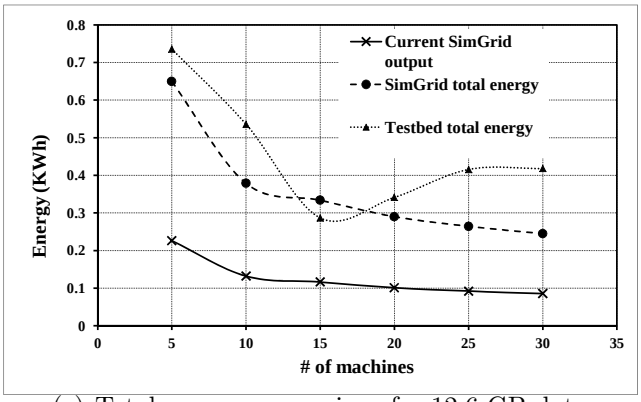
that, computation power means the power which is necessary to finish the given task by the participant machines. In the case of *SimGrid*, computation power can be measured by the existing energy plug-in. In this thesis, we use computation power and energy consumption (without cooling energy) interchangeably.



(a) Cooling energy comparison for 12.6 GB data



(b) Computation energy comparison for 12.6 GB data

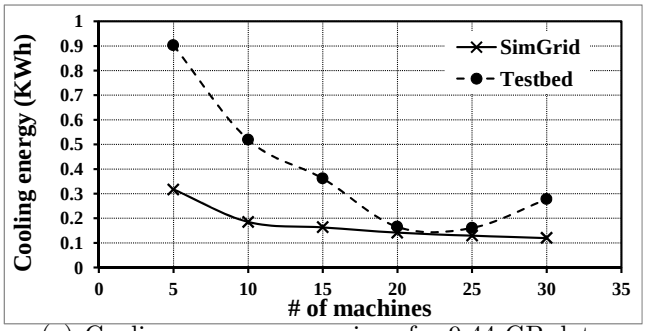


(c) Total energy comparison for 12.6 GB data

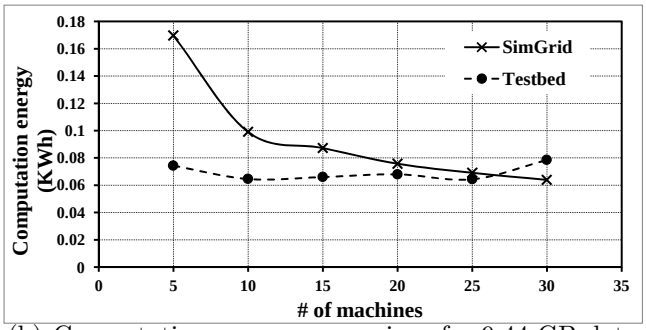
Figure 5.4: Energy comparison between *SimGrid* and Testbed for 12.6 GB data

Table 5.3: Average and standard deviation comparison between testbed and *SimGrid*

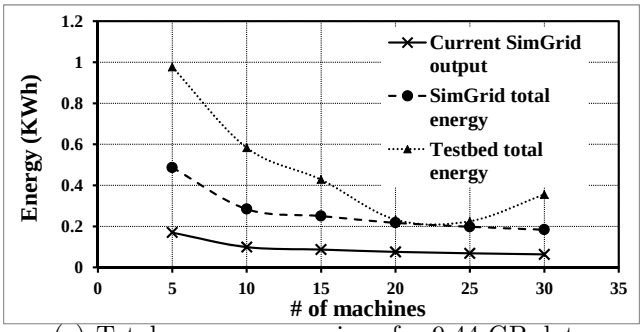
Workload	Cooling energy consumption(KWh)				Computational energy consumption(KWh)				% deviation in *SimGrid* w.r.t. testbed			
	Average		Std. deviation		Average		Std. deviation		Cooling energy		Computational energy	
	Testbed	*SimGrid*	Testbed	*SimGrid*	Testbed	*SimGrid*	Testbed	*SimGrid*	Average	Std. deviation	Average	Std. deviation
12.6 GB	0.367	0.235	0.156	.0975	0.088	0.125	0.006	0.052	36.1	37.7	41.6	766.7
9.44 GB	0.398	0.176	0.281	.073	0.069	0.094	0.006	0.039	55.7	74.0	35.6	550.0
6.3 GB	0.242	0.117	0.134	0.049	0.052	0.063	0.006	0.026	51.6	63.4	19.9	348.3
3.14 GB	0.146	0.058	0.058	0.025	0.030	0.031	0.002	0.013	60.2	48.6	5.1	500.0



(a) Cooling energy comparison for 9.44 GB data



(b) Computation energy comparison for 9.44 GB data



(c) Total energy comparison for 9.44 GB data

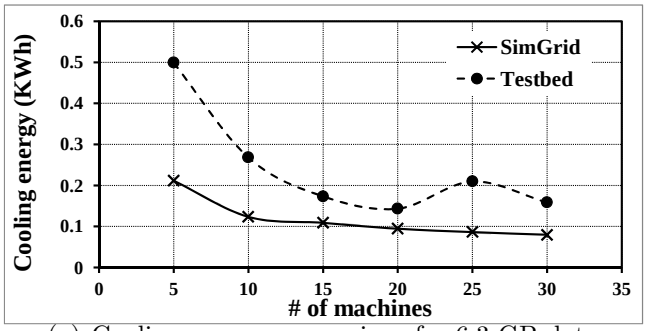
Figure 5.5: Energy comparison between *SimGrid* and Testbed for 9.44 GB data

5.3.2 Testbed Compatible Settings in *SimGrid*:

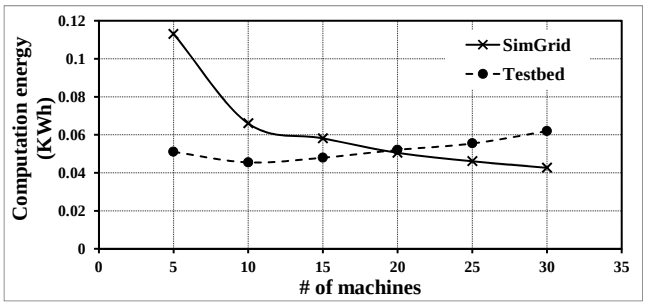
After setting up the real testbed, we try to imitate the lab environment into our *SimGrid* environement. To imitate the testbed, we need to make some conversion over parameter units as *SimGrid* uses a different unit system for measuring machine power. *SimGrid* uses FLOating Points per Second (FLOPS) unit for evaluating the power of machines. We use the following formula to evaluate FLOPS [54] [55] [56]:

$FLOPS = N_s \times \frac{N_c}{N_s} \times \frac{Cycles}{second} \times \frac{FLOPS}{Cycle}$

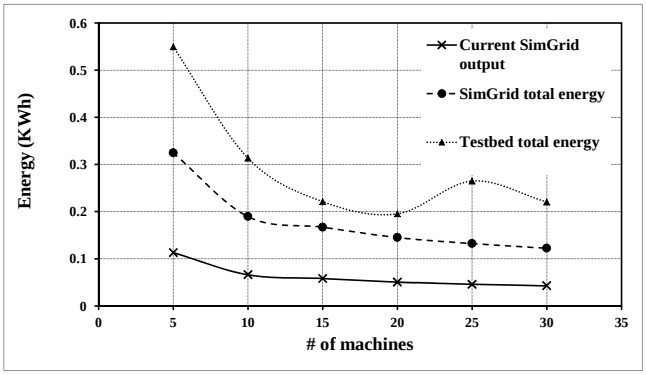
Here, N_s refers to the number of sockets, N_c refers to the number of cores per socket, $\frac{Cycles}{second}$



(a) Cooling energy comparison for 6.3 GB data



(b) Computation energy comparison for 6.3 GB data



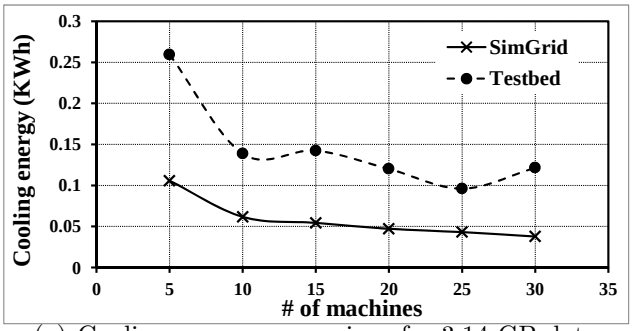
(c) Total energy comparison for 6.3 GB data

Figure 5.6: Energy comparison between *SimGrid* and Testbed for 6.3 GB data

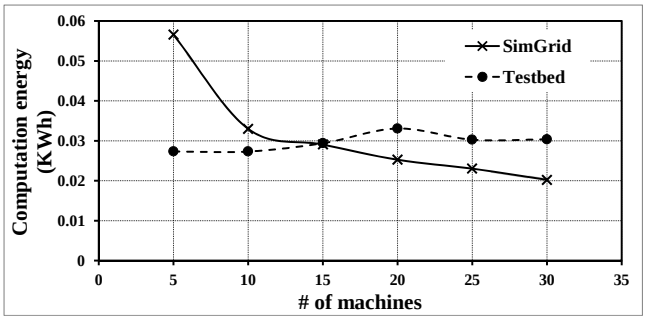
refers to the frequency of the machine, and $\frac{FLOPS}{Cycle}$ refers to the floating point operations per cycle.

We adopt these values from the specifications of the laboratory machines. Table 5.5 shows the simulation environment.

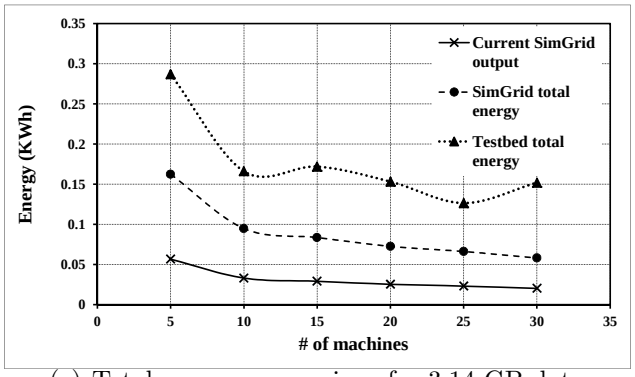
We create each file using random texts of 787MB size. We use different number of files ranging from 4 to 16. Hence, we consider data size from 3.14GB to 12.6GB. In *SimGrid*, we have to give the number of chunks and the size of each chunk. We divide our total data size



(a) Cooling energy comparison for 3.14 GB data



(b) Computation energy comparison for 3.14 GB data



(c) Total energy comparison for 3.14 GB data

Figure 5.7: Energy comparison between *SimGrid* and Testbed for 3.14 GB data

into five thousand chunks. For example, 12.6GB data will be divided into equal sized five thousand chunks. This is a random value, however, other chunk sizes show similar results. We know from our early description that each node can have three power states. From our testbed experiment, we get most nodes consume 100W-120W power. The more the load, the more the power consumption. It is also known from our testbed experiment that when the machine is in idle state it dissipates 40W power. At power-off state, we have only 5W power consumption. Table 5.4 shows the power states. Fig. 5.8 shows how we measure cooling power and computation power. *SimGrid* generally divides a large file into small chunks similar

Table 5.4: Different power states and consumed power

Power state	Consumed power
Active state	100W-120W
Idle state	40W
Power-off state	5W

Table 5.5: Simulation environment in SimGrid

Parameter	Value
# of master machines	1
# of slave machines	4, 9, 14, 19, 24, 29
PC power	38,400-44,800 Mega FLOPS
PC power consumption	Peak: 100 - 120 W, idle: 40 W, power off: 5 W
Line bandwidth	100 kbps
Total # of files	4, 8, 12, 16
Size of each file	787 MB
Total data size	3.14. 6.3, 9.44, 12.6 GB
Maximum allowable temperature	22° C
Environment temperature	25° C

to Hadoop. Hence, if we use big files (more than 787MB) then the number of chunks will be increased, and the simulation will take much longer time to finish and will consume more cooling energy.

5.3.3 Experimental Results

We create an identical environment, and distribute same amount of work among nodes. We experiment the whole process similarly in both *SimGrid* and testbed platform. After that, we compare cooling energy, computation energy, and total energy between testbed and *SimGrid*. We can see the comparison from Fig. 5.4-5.7 for various data sizes. For each data size, these figures show the comparison for cooling energy, computation energy, and total energy. If we increase the number of machines, the energy gets decreased, and this is evident in all the graphs. It is visible from the fact that all the graphs have a downward slope. If we use smaller number of machines then all the machines need to work for a long time. At the same time, the overall power consumption gets decreased as we use smaller number of machines. However, the ratio of increasing computation time is much more greater than the decrease in power consumption. As we get energy from multiplying computation time with consumed power,

the overall energy consumption gets decreased with the increase in number of machines. On the other hand, if we use more number of machines, the machines need to work for a smaller amount of time and decrease consumed energy.

SimGrid previously had computation energy module. The comparison of computation energy between existing *SimGrid* and real testbed can be seen from Fig. 5.4(b). 5.5(b), 5.6(b), and 5.7(b). We can see a common pattern of decreasing computation energy as we increase the number of machines. Moreover, when the number of machines is 5, the difference between *SimGrid* and real testbed is bigger than other differences. *SimGrid* and real testbed have almost equal value when there are 15-20 machines.

We can see the cooling energy comparison graphs in Fig. 5.4(a). 5.5(a), 5.6(a), and 5.7(a). We can see, cooling energy comparison graphs follow a similar pattern as the existing computation energy comparison graphs. These graphs also show a big difference when the number of machines is 5, and show nearly same cooling energy when there are 15-20 machines. Hence, we can say that our modeled cooling energy curves follow the similar pattern of the existing computation energy curves.

We can see the total energy comparison from Fig. 5.4(c). 5.5(c), 5.6(c), and 5.7(c). "Current *SimGrid* output" line shows the total energy value if we do not integrate the cooling energy. *SimGrid* line shows the total energy after integrating cooling energy, and testbed total energy line shows the total energy for testbed. We can see from these graphs that if we do not integrate cooling energy, then the total energy value is far less than the actual value of the total energy. After integrating cooling energy, we can get a close total energy line between *SimGrid* and real testbed.

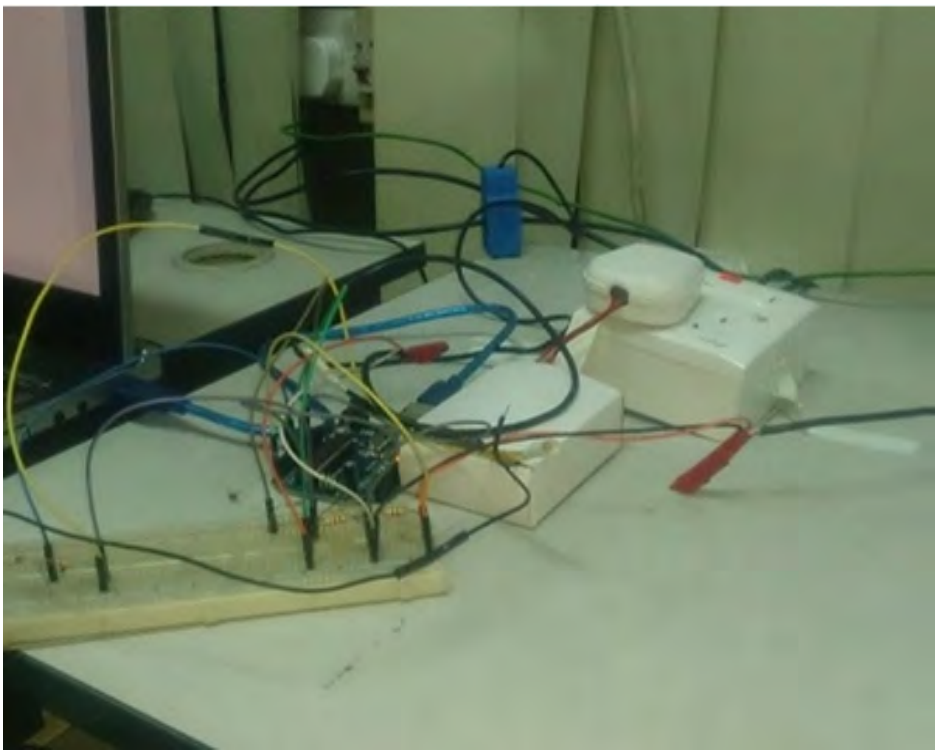
Both the results from *SimGrid* and real testbed have some variations, however, they are closely similar. If we do not integrate cooling energy with *SimGrid*, this difference will be more prominent.

In Table 6.2, we show the comparison of average value and standard deviation between testbed and *SimGrid*. We present the result for various workloads.

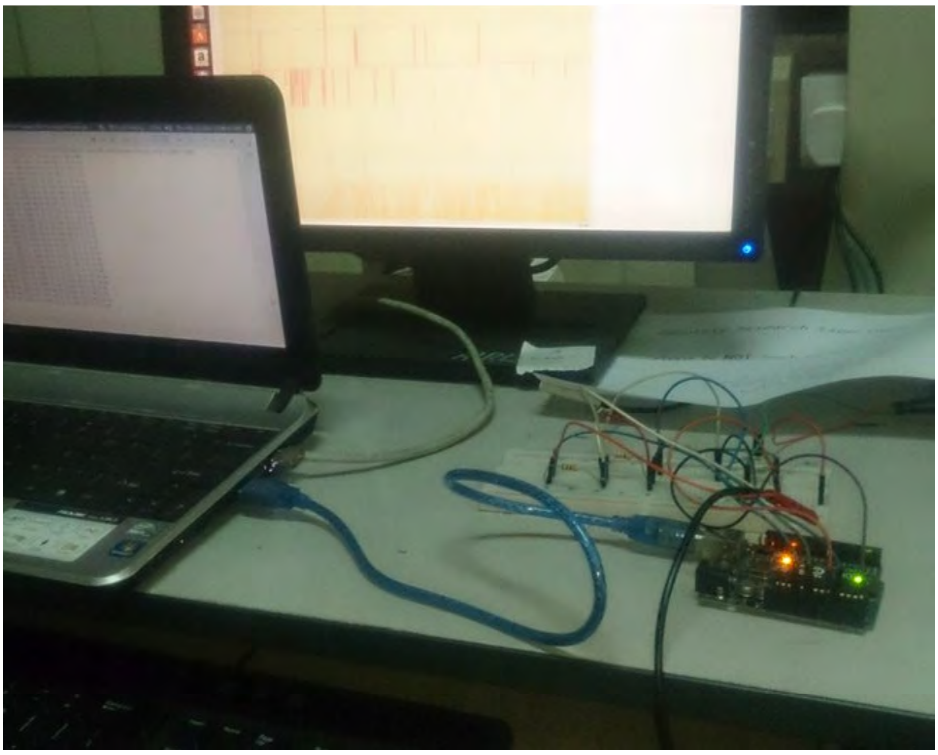
5.3.4 Summary of Findings

We can summarize our findings as follows:

- Both the computation energy and cooling energy deviate similarly from the testbed energy consumption. Hence, our module for cooling energy is compatible with the existing energy model of *SimGrid*.
- Number of machines and energy consumption follow an inverse relationship. This inverse relationship is observable in our cooling energy model as well as existing computation energy model. Testbed result also validates this.
- In most cases, if we increase the workload then the consumed energy is increased. We get the similar trend for *SimGrid* simulation and testbed experiment, though there are some exceptions.
- Integrating cooling energy is necessary to get a closer total energy consumption to real testbed.



(a) Measuring computation power



(b) Measuring cooling power

Figure 5.8: Measuring computation and cooling power

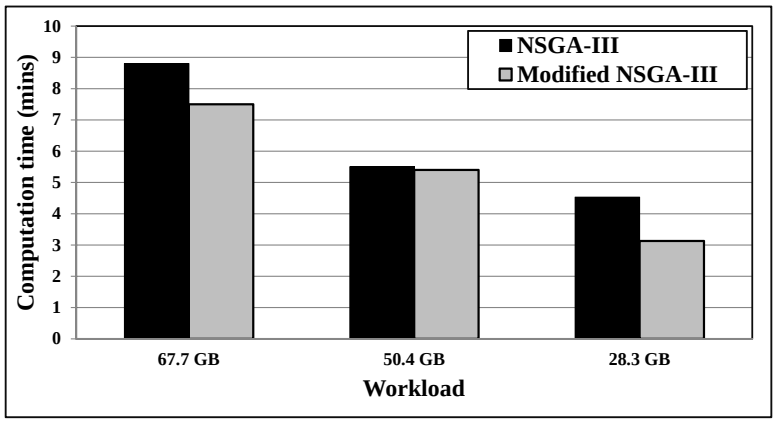
Chapter 6

Experimental Results

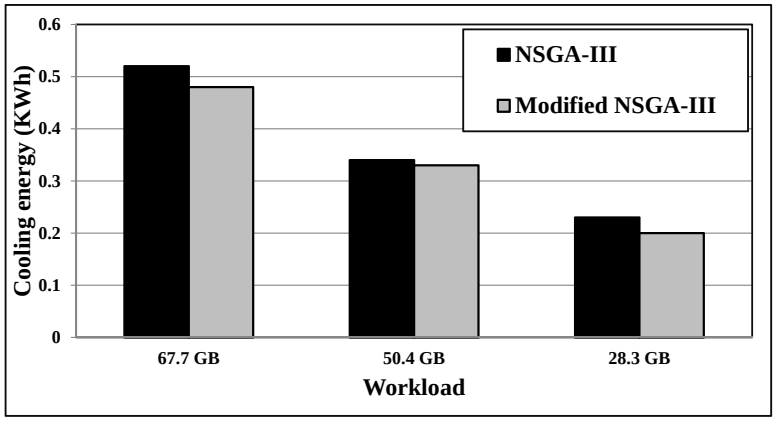
In this chapter, we will show the experimental results in both platform. We rigorously experimented our approach with other existing approaches in both real testbed and simulation platform *SimGrid*. In Chapter 4, we see the solution approach to solve the problem indicated in Chapter 3. We implemented an NSGA-III based algorithm. Hence, at first, we will see how our modified NSGA-III algorithm performs better than the existing NSGA-III algorithm. Then we will discuss our experimental results and present a comparative results among our approach and existing other approaches.

6.1 Validation Modified NSGA-III Performs Better Than NSGA-III

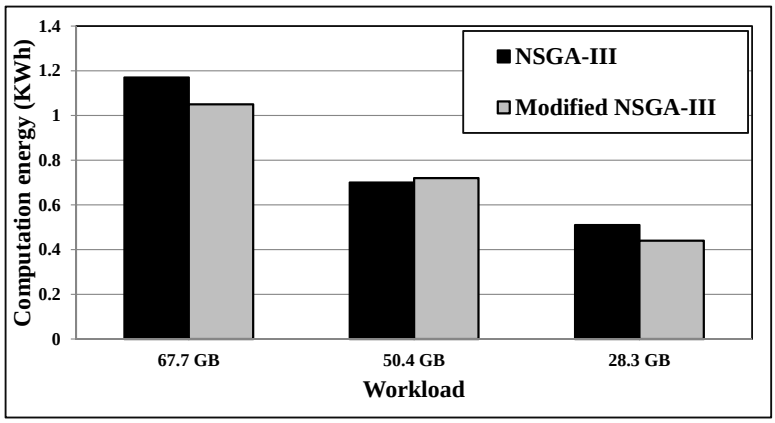
We use Table 6.1 for showing the modified NSGA-III technique performs better than the existing NSGA-III approach. We use different workloads from 28.3 GB to 67.7 GB. We use 30 machines cluster in this case. Fig. 6.1 shows the experimental results in *SimGrid*. From Fig 6.1(a), we can see the comparison for computation time. For 28.3 GB and 67.7 GB data, modified NSGA-III performs better than the existing NSGA-III algorithm. For 50.4 GB data, both NSGA-III and modified NSGA-III perform similarly. In Fig. 6.1(b), we can see the comparison for cooling energy consumption in different workloads. For each workload, we can see that modified NSGA-III performs better than the NSGA-III though for 50.4 GB data they are very



(a) Computation time comparison



(b) Cooling energy consumption comparison



(c) Computation energy consumption comparison

Figure 6.1: Comparison of NSGA-III and modified NSGA-III with various workloads in *SimGrid* with 30 machines cluster

close to each other. From Fig. 6.1(c), we can see another comparative result for computation energy. Here, modified NSGA-III performs better than the NSGA-III for data sizes 28.3 GB

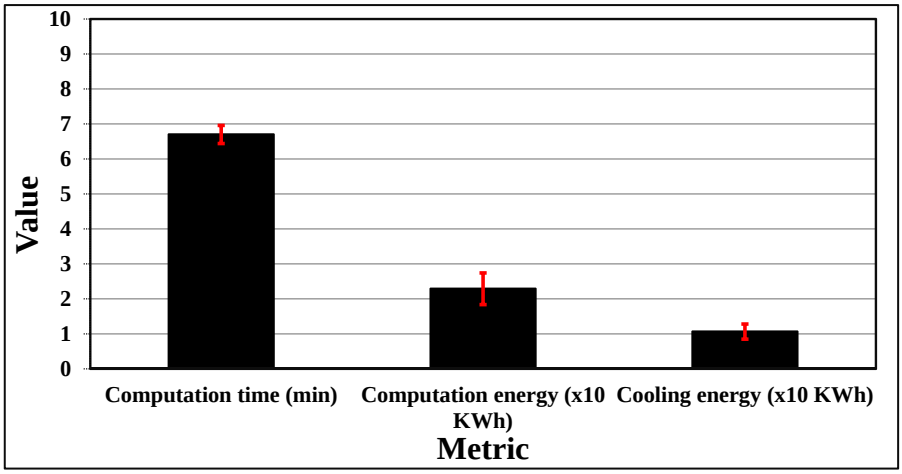


Figure 6.2: Average of computation time, computation energy, and cooling energy in various iterations with standard deviations

and 67.7 GB. NSGA-III performs better in this case for 50.4 GB data.

In real testbed, we perform the analysis for 31.5 GB data. We found NSGA-III needs 26 minutes 18 seconds to finish the task, however, modified NSGA-III needs 25 minutes 21 seconds to finish the task. Hence, modified approach performs better than the NSGA-III. Modified NSGA-III consumed slightly less cooling energy and computation energy too. Modified NSGA-III needs 2.07 KWh energy, whereas, NSGA-III needs 2.15 KWh cooling energy. For computation energy, modified NSGA-III needs 0.23 KWh energy and NSGA-III needs 0.25 KWh computation energy. NSGA-III has some improved result for CPU and memory utilization. It utilizes 74.7% CPU and 91.3% memory utilization. Modified NSGA-III, on the other hand, utilizes 72.0% CPU and 91.0% memory. As in most cases, modified NSGA-III performs better we use modified NSGA-III for solving the many-objective optimization problem. Next we will discuss about the simulation experimentation and then we will discuss about the experimental results for real testbed.

6.2 Effects of Number of Iterations over Performance

For a single setting, we run our NSGA-III algorithm for 10 times. We get different outputs. However, as we use a greedy approach, there are some machines which are selected more often than the other machines. At the same time, there are some machines which are selected less

Table 6.1: Simulation environment in SimGrid

Parameter	Value
# of master machines	1
# of slave machines	29, 49
PC power	4,000-38,000 Mega FLOPS
PC power consumption	Peak: 10 - 400 W, idle: 2.5-100 W, power off: 5 W
Line bandwidth	10-100 kbps
Total # of files	86, 64 and 36
Size of each file	787 MB
Total data size	67.7 GB, 50.4 GB and 28.3 GB
Maximum allowable temperature	$20 - 23^\circ$ C
Environment temperature	28° C
SimGrid version	3.12

than the other machines. Hence, over the solutions in different iterations we find a common pattern of selected machines. That is why when we evaluate performance we get a similar performance pattern over different iterations. From Fig. 6.2, we can see the average and standard deviation for various iterations. In most cases, we get performance values close to the average value as we select similar type of machines in each period. We use probabilistic conditions to select the machines after clustering and identifying the best or worst machine. This probability helps us not to select the biased machines all the time.

6.3 Simulation Evaluation

We test our algorithm and other existing techniques in a simulation environment. We use 30 and 50 machines clusters to test our algorithm and compare other techniques with our technique. We use computation time, cooling energy, and simulation energy as our comparative parameters. We cannot get CPU utilization and memory utilization directly from *SimGrid*. Hence, we ignore these utilizations. We present utilization-based comparison in real testbed results.

6.3.1 Simulation Settings

For simulation we use a well-known simulation tool for distributed system named *SimGrid* [13]. Our modified *SimGrid* has energy plug-in, which can compute both computation and cooling

energy [57]. Table 6.1 shows the simulation environment in *SimGrid*. *SimGrid* uses different unit conventions while setting-up machines in clusters. The conversion between *SimGrid* unit and traditional unit can be found in Chapter 5. We use one master machine, and 29, 49 slave machines. In *SimGrid* each machine should be given a power value. We use 4000-38000 Mega FLOPs for giving the machine power values. We try to keep this value close to the real testbed setup machine properties. We have three different level of power consumption for the machines of *SimGrid*. We use up-to 400W when the machines are in full load, we use 2.5-100W when the machines are idle, and we use 5W power consumption when the machines are off. We use line bandwidth from 10 kbps to 100 kbps. There are 86, 64, and 36 files for which we implemented the typical word count job. Each data file had 787 MB of data making total data size of 67.7 GB, 50.4 GB, and 28.3 GB data respectively. We use 5° C difference in temperature. We use *SimGrid 3.12* version for our simulation.

6.3.2 Simulation Results

Figures from 6.3(a) to 6.3(d) show the simulation results in *SimGrid* for 30 machines and figures from 6.4(a) to 6.4(d) show the simulation results for 50 machines. We have the results for 67.7 GB, 50.4 GB, and 28.3 GB data. White bar indicates the results for modified NSGA-III, gray line indicates the result for Ant Colony Optimization (ACO) technique, and black line indicates the simulation results for Particle Swarm Optimization (PSO) technique. In most of the comparison, white bar is below than the other two bars which indicate the efficacy of our proposed algorithm. Table 6.2 and 6.3 show the improvement made by our technique for 30 and 50 machines respectively.

6.3.3 Findings from Simulation Results

We test our algorithm with different workloads using the simulation environment of Table 6.1. Table 6.2 and 6.3 show the comparison. We take the following parameters as the cluster objectives:

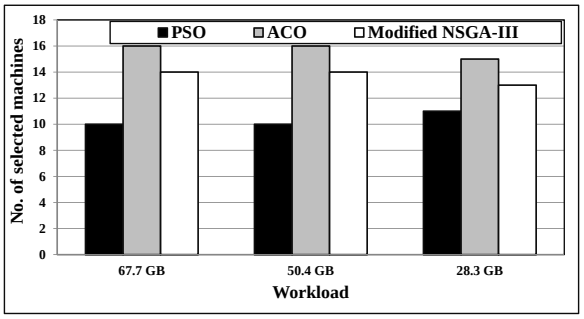
- Computation time
- Cooling energy

Workload	Improvement over PSO			Improvement over ACO		
	Time(%)	Cooling energy (%)	Comp. energy (%)	Time (%)	Cooling energy (%)	Comp. energy (%)
67.7 GB	21	13	10	43	10	5
50.4 GB	36	11	10	17	8	8
28.3 GB	43	13	10	15	5	0

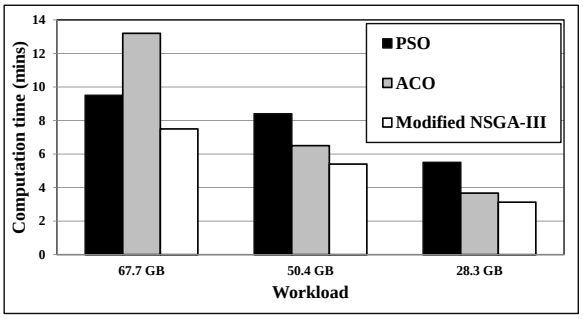
Table 6.2: Improvement over PSO and ant colony optimization with various workloads in *SimGrid* with 30 machines

- Computation energy

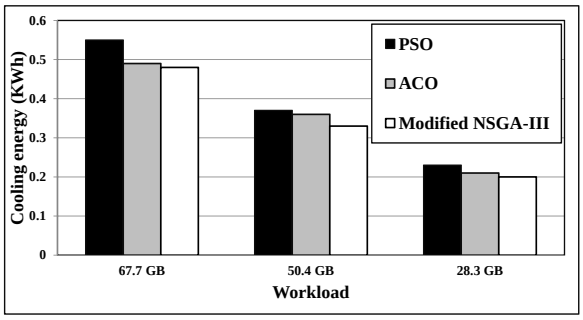
We use 67.7 GB, 50.4 GB and 28.3 GB workloads and consider 30 and 50 machines clusters. We can see from Table 6.2 and 6.3 that modified NSGA-III gives better performance than other methods for all three objectives in most cases. In Table 6.2, we show the improvement made by our modified NSGA-III algorithm over PSO and ant colony optimization technique for 30 machines. We can see the computation time is improved by 21%, 36% and 43% than PSO and we have 43%, 17% and 15% improvement over ant colony optimization with different workloads. Cooling energy is improved by 13%, 11% and 13.0% than PSO. It is also improved by 10%, 8% and 5% than ant colony optimization. Computation energy is also improved by the modified NSGA-III algorithm. We have nearly 10% improvement over PSO and nearly 5% to 8% improvement over ant colony optimization. We also show the performance comparison for 50 machines in Table 6.3. Here, we have 10%-33% improvement over PSO for computation time. Modified NSGA-III also has up-to 59% improvement for computation and cooling energy over PSO. We have some negative values for energy also. Evolutionary approaches cannot guarantee to get the optimal values all the time. However, in most cases, modified NSGA-III performs better than the other existing approaches. Over ACO, modified NSGA-III gives improvement within 16% to 46% for computation time. We have 55% to 87% improvement for both cooling energy and computation energy. Now, we will show the experimental results for real testbed.



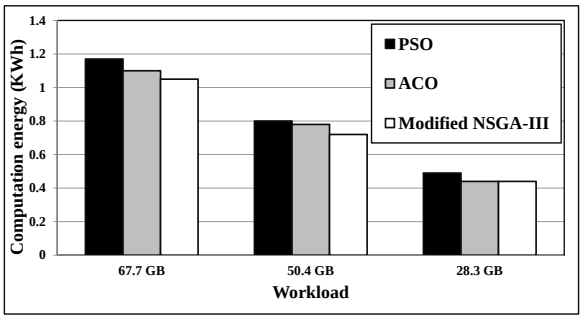
(a) No. of selected machines comparison



(b) Computation time comparison



(c) Cooling energy consumption comparison



(d) Computation energy consumption comparison

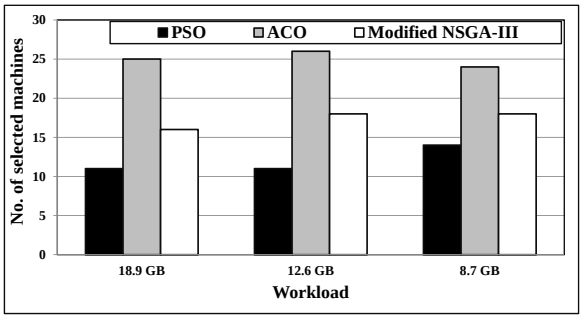
Figure 6.3: Comparison of different algorithms with various workloads in *SimGrid* with 30 machines cluster

Workload	Improvement over PSO			Improvement over ACO		
	Time(%)	Cooling energy (%)	Comp. energy (%)	Time (%)	Cooling energy (%)	Comp. energy (%)
67.7 GB	38	59	59	16	55	55
50.4 GB	33	36	38	41	79	79
28.3 GB	10	-6	0	46	87	87

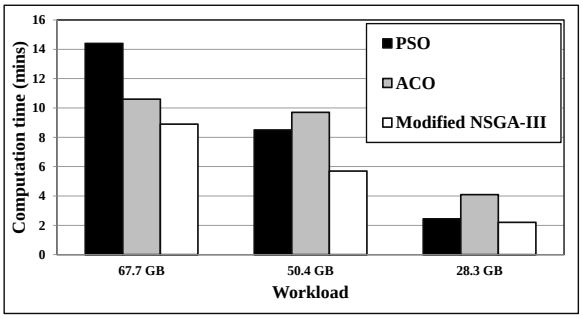
Table 6.3: Improvement over PSO and ant colony optimization with various workloads in *SimGrid* with 50 machines

6.4 Testbed Evaluation

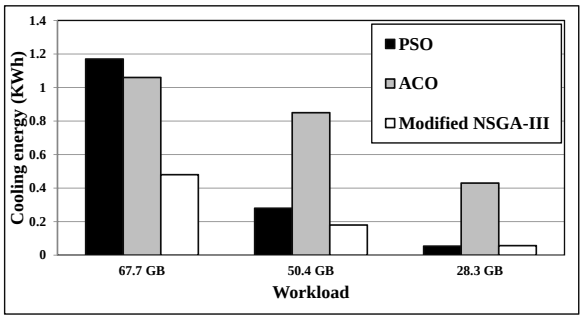
We test our algorithm and other existing techniques in a real testbed environment. We run our modified NSGA-III algorithm before implementing any workload into the cluster. After selecting the machines we only power on the selected machines. All other machines are in sleep state. We continuously log our data for each slave machines. We have some sensors too to measure different current and voltage to measure the power. Similarly, we also implement other approaches like ACO and PSO. We test these two techniques also in our laboratory environment.



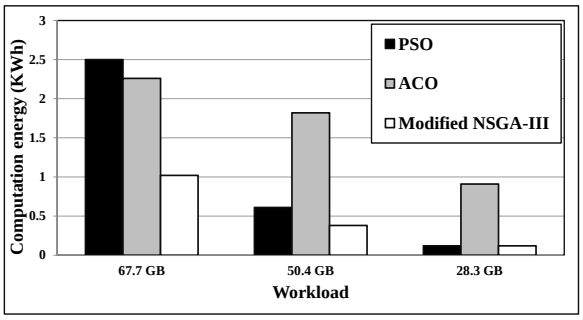
(a) No. of selected machines comparison



(b) Computation time comparison



(c) Cooling energy consumption comparison



(d) Computation energy consumption comparison

Figure 6.4: Comparison of different algorithms with various workloads in *SimGrid* with 50 machines cluster

6.4.1 Testbed Settings

Description of the testbed setup was presented in Chapter 5. We are also presenting the environment here as we made some changes in the configuration. Table 6.4 and 6.5 show the testbed configuration. We use 30 machines cluster to test our algorithm and compare other techniques with our technique. In this case also, we use one master machine and our cluster has 29 slave machines. We have different CPU frequency ranging from 2.4 GB to 2.8 GB. We have memory range from 1 GB to 2 GB. We use 5 to 100 MBPS. We have Ubuntu 14.04 LTS OS in all the machines.

Fig. 6.5 shows how we map our machines in the laboratory. We test our technique in a heterogeneous environment to see the effect of our greedy and NSGA-III technique. To create heterogeneity, we use *Wondershaper* tool to control the network bandwidth. Table 6.6 shows the network bandwidth of various machines. Note that, this is not the value that a machine could get. The table data is 8-10 times than the original value. *SimGrid* and real testbed use different unit conventions. Hence, imitating the real testbed into *SimGrid* is not completely

Table 6.4: Experimental environment in laboratory

Parameter	**Value**
No. of master	1
No. of slaves	29
Processor	Intel Core 2 Duo
Processor base frequency	2.4 GHz, 2.66 GHz and 2.8 GHz
Memory	1 GB to 2 GB
Network B/W	5 to 100 MBPS
OS	Ubuntu 14.04 LTS (x86)
Total number of files	40, 24, 16, 11
Size of each file	787 MB
Total data size (custom made)	31.5 GB, 18.9 GB, 12.6 GB, and 8.7 GB
Total data size (benchmark)	26 GB (1 file), 1 GB (1 file)

Table 6.5: Frequency and memory vs number of machines

Processor base frequency	**Memory**	**No. of machines**
2.4 GHz	1 GB	1
2.4 GHz	2 GB	2
2.66 GHz	1 GB	2
2.66 GHz	2 GB	8
2.8 GHz	2 GB	17

possible. Besides, we use different performance metrics for real testbed and simulation environment. In real testbed, we can get resource utilization, however, in the existing *SimGrid* we cannot get resource utilization.

6.4.2 Testbed Results

We can see the testbed results in Figures 6.7(a) to 6.7(f). We use different workloads and consider the following five cluster objectives:

- Computation time
- Cooling energy
- Computation energy
- CPU usage
- Memory usage

Table 6.6: Machine ID with corresponding network B/W

Machine ID	**Network B/W (mbps)**
0	18
1	35
2	50
3	40
4	27
5	20
6	65
7	25
8	100
9	100
10	30
11	60
12	49
13	40
14	8
15	69
16	13
17	19
18	92
19	13
20	26
21	100
22	5
23	100
24	5
25	100
26	42
27	75
28	10
29	37

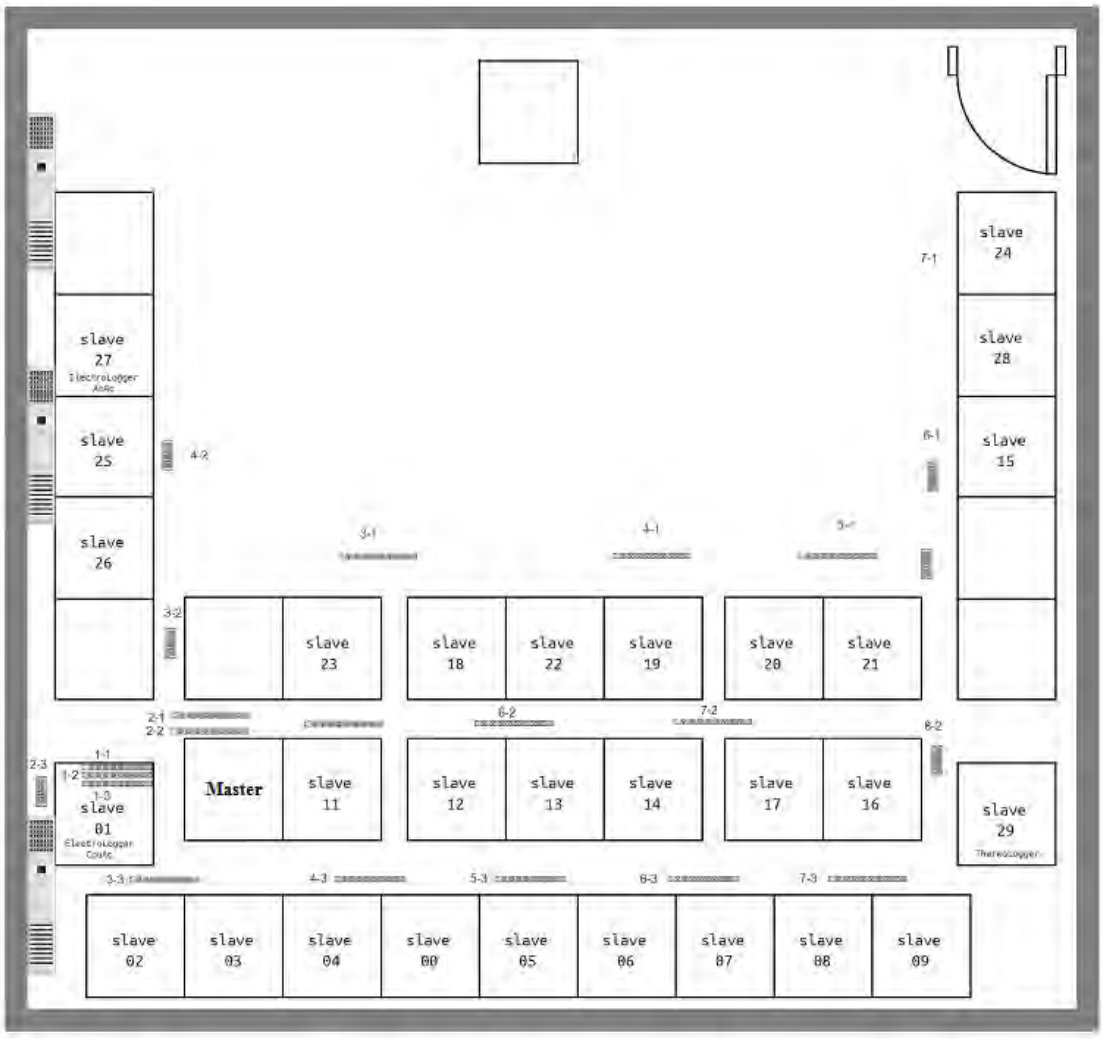
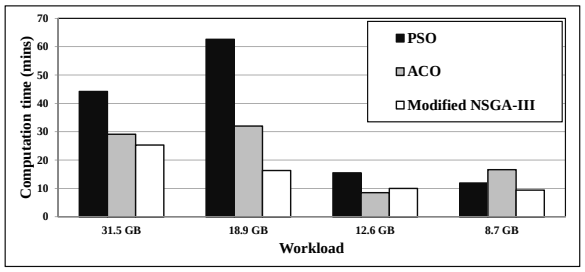


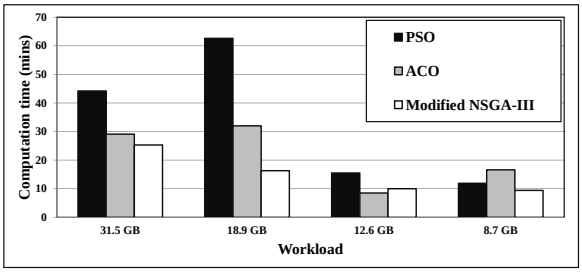
Figure 6.5: Lab map for cluster

Table 6.7: Improvement over PSO and ant colony optimization with various workloads in real testbed

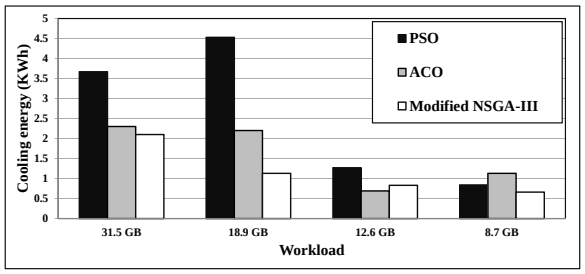
Workload	**Improvement over PSO**					**Improvement over ACO**				
	Time(%)	**Cooling energy (%)**	**Comp. energy (%)**	**CPU usage (%)**	**Mem. usage (%)**	**Time(%)**	**Cooling energy (%)**	**Comp. energy (%)**	**CPU usage (%)**	**Mem. usage (%)**
31.5 GB (custom)	43	75	22	1	5	13	49	28	23	4
26 GB (benchmark)	20	6	9	4	1	-16	-17	4	23	1
18.9 GB (custom)	74	75	63	127	5	49	49	58	159	0
12.6 GB (custom)	36	35	25	0	-1	-18	-20	10	27	3
8.7 GB (custom)	21	21	20	10	-7	43	42	56	130	1
1 GB (benchmark)	4	7	-13	4	0	0	21	-2	65	0



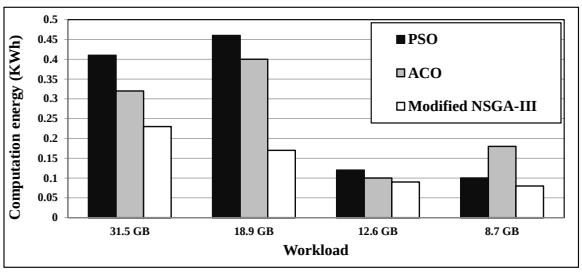
(a) No. of selected machines comparison



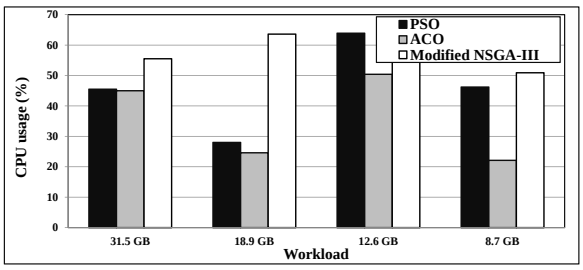
(b) Computation time comparison



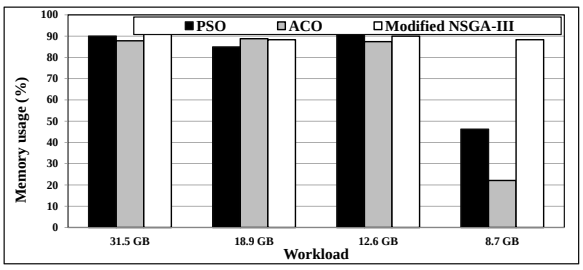
(c) Cooling energy consumption comparison



(d) Computation energy consumption comparison



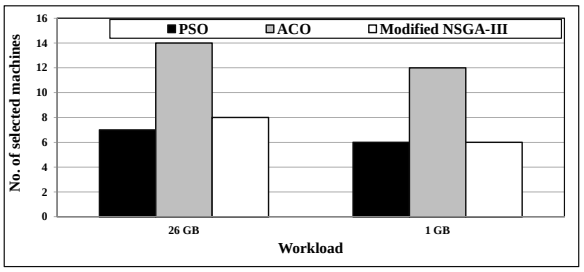
(e) CPU usage comparison



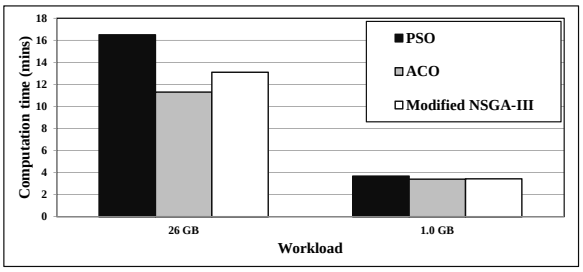
(f) Memory usage comparison

Figure 6.6: Comparison of different algorithms with various custom made workloads in real testbed with 30 machines cluster

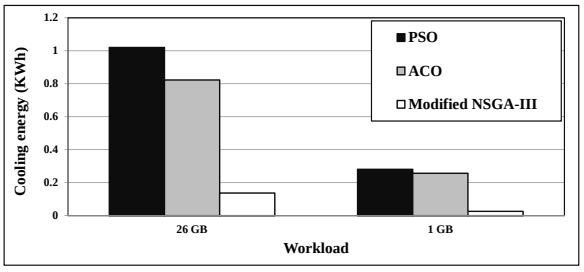
Along X axis, we take the workloads. We experimented with 8.7 GB, 12.6 GB, 18.9, and 31.5 GB of data. We also use benchmark data of 1 GB and 26 GB size [58]. In all the cases, white box indicates our modified NSGA-III approach, gray box indicates ACO, and black box indicates PSO technique. We can see from these graphs that, in most cases, white bars show better performance than the other two bars. We show all the comparisons among the techniques in Table 6.7.



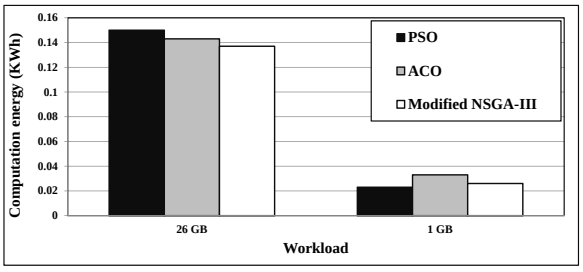
(a) No. of selected machines comparison



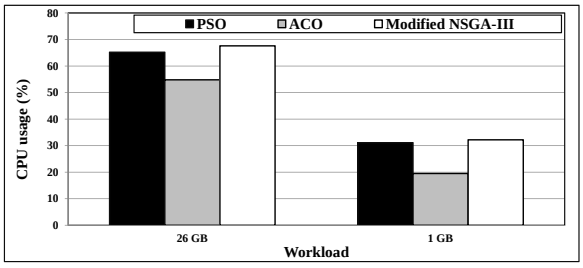
(b) Computation time comparison



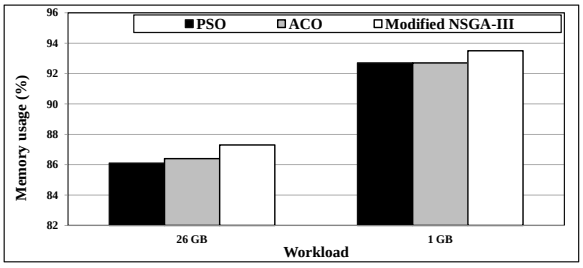
(c) Cooling energy consumption comparison



(d) Computation energy consumption comparison



(e) CPU usage comparison



(f) Memory usage comparison

Figure 6.7: Comparison of different algorithms with various benchmark workloads in real testbed with 30 machines cluster

6.4.3 Findings from Testbed Results

We compare our technique and other existing techniques in the real testbed mentioned in Figures 6.7(a) to 6.7(f) and in Table 6.7. As we previously mentioned, we take number of machines, computation time, computation energy, cooling energy, CPU usage and memory usage as our cluster objectives. We want to decrease the number of machines (to decrease cost), computation time, computation energy and cooling energy. At the same time, we want to increase the CPU and memory usage. From Table 6.7, we can see the comparison. With 31.5 GB data, NSGA-III performs better than the other three approaches. For the case of 18.9 GB

workload, NSGA-III shows significant improvement than the other two techniques for all the objectives. With 12.6 GB data, NGSA-III performs better for most of the cluster objectives. For 8.7 GB workload, though PSO has a better memory usage, NSGA-III performs better for all other objectives. For 31.5 GB data we have 43%, 75%, 22%, 1%, and 5% improvement over PSO and 13%, 49%, 28%, 23%, and 4% improvement over ACO. In the case of 18.9 GB data, we have 49% to 74% improvement over computation time, 49% to 75% improvement over cooling energy, 58% to 63% improvement over computation energy, 127% to 159% improvement over CPU usage and 0% to 5% improvement over memory usage. In the case of 12.6 GB data, we have up-to 36% improvement over computation time, 35% improvement over cooling energy, 25% improvement over computation energy, 27% improvement over CPU usage and 3% improvement over memory usage. For 8.7 GB data, we get upto 43% improvement over PSO, 42% improvement over cooling energy, 56% improvement over computation energy, 130% improvement over CPU usage, and 1% improvement over memory usage. For benchmark data, with 26 GB data modified NSGA-III has 20% improvement in computation time, 6% improvement in cooling energy, 9% improvement in computation energy, 4% improvement in CPU usage, and 1% improvement in memory usage over PSO. Modified NSGA-III has no improvement in computation time and cooling energy, 4% improvement in computation energy, 23% improvement in CPU usage, and 1% improvement in memory usage over ACO. For 1 GB data, we have improvement in computation time, cooling energy, and CPU usage over PSO. We also have improvement in cooling energy and CPU usage over ACO.

6.5 Improvement over PSO and ACO in Real Testbed and *SimGrid*

We have close improvements over PSO and ACO in real testbed and *SimGrid* on an average. Fig. 6.8 shows the results. Here, gray box indicates percentage of improvement over PSO and ACO in *SimGrid*, and black box indicates percentage of improvement over PSO and ACO in real testbed. On an average, we can see both graphs are close to each other. Hence, we have similar type of improvements in *SimGrid* and real testbed which indicates the robustness of our approach. For each bar, we also show the error bars which was drawn by standard deviations.

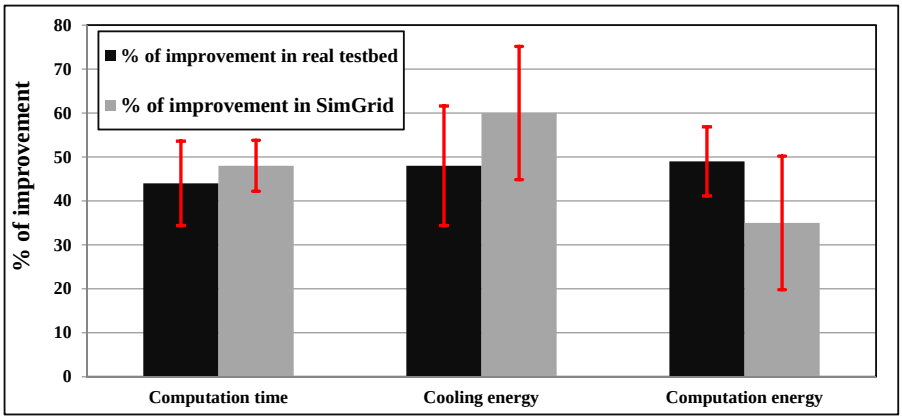


Figure 6.8: Improvement over PSO and ACO in Real Testbed and SimGrid

This helps us to understand what could be the lowest and highest values, and how they differ from the average value.

Chapter 7

Conclusion and Future Work

Solving conflicting objectives in clusters to provide administrators a set of machines is important, and only a few studies attempt to solve this problem for computing clusters. Moreover, the existing studies focus on optimizing single or multi-objective perspectives. On the contrary, in this thesis, we provide a many-objective solution for cluster administrators to select the cluster nodes. Here, we exploit a synergy between a greedy technique and NSGA-III algorithm to solve the many-objective problem. We test our technique with other existing techniques and show our proposed technique reveals the best set of nodes in most of the cases, which fulfill the cluster objectives. We experiment in a real testbed, which confirms robustness and efficacy of our technique. We also simulate our technique in a simulation platform named *SimGrid* to test in larger clusters. We add a cooling energy module into the existing *SimGrid*. To augment all these testing, we collect real data for a period of more than one year and conduct empirical analyses over the data. We consider different environmental settings in different seasons of the year to ensure robustness of the empirical analyses. Our technique engendered from the analyses can be used in real clusters to select the best combination of machines, which will fulfill the cluster objectives.

In our study, we use only one laboratory environment to experiment real testbed setup. In future, we need to deploy our model in different laboratory environments. We use NSGA-III as our base algorithm, as literature shows NSGA-III works well with many-objective optimization problems. Further experimental study is necessary to prove it in cluster environment. We also need to experiment with more number of machines in real testbed. There are many other

factors which have impacts on cooling energy. Among them room size, room cooling capacity, environment temperature are the most important ones. We need to incorporate those factors to get a more robust results. We only consider the word-count job, hence, different type of jobs should be tested with the discussed approaches. Moreover, we should test our experiment with various types of content.

Bibliography

[1] B. Barney, "Introduction to Parallel Computing." `https://computing.llnl.gov/tutorials/parallel_comp/`, 2017. [Online; accessed 29-June-2017].

[2] Mason and Hanger, *Data Center Energy Consumption*, Dec 2015. Available online at `http://www.ha-inc.com/blog/entry/data-center-energy-consumption/`. Last accessed April 1, 2017.

[3] K. Xiong and S. Suh, "Resource provisioning in sla-based cluster computing," in *Workshop on JSSPP*, pp. 1–15, Springer, 2010.

[4] T. White, *Hadoop: The Definitive Guide*. O'Reilly, first edition ed., june 2009.

[5] S. Wadkar, M. Siddalingaiah, and J. Venner, *Pro Apache Hadoop*. Apress, 2014.

[6] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[7] G. Singh, C. Kesselman, and E. Deelman, "A provisioning model and its comparison with best-effort for performance-cost optimization in grids," in *Proceedings of the* 16^{th} *HPDC*, pp. 117–126, ACM, 2007.

[8] K. Tarplee, A. Maciejewski, and H. Siegel, "Robust performance-based resource provisioning using a steady-state model for multi-objective stochastic programming," *IEEE Transactions on Cloud Computing*, 2016.

[9] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints.," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[10] X. Wang, Y. Wang, and Y. Cui, "A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing," *Future Generation Comp. Syst.*, vol. 36, pp. 91–101, 2014.

[11] "The Network Simulator NS-2." `http://www.isi.edu/nsnam/ns/`. [Online; accessed 15-August-2016].

[12] "The Network Simulator NS-3." `https://www.nsnam.org/`. [Online; accessed 15-August-2016].

[13] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Simgrid: a sustained effort for the versatile simulation of large scale distributed systems," *CoRR*, vol. abs/1309.1630, 2013.

[14] Chaos, "Difference among clusters, clouds, and grids." `https://stackoverflow.com/questions/9723040/what-is-the-difference-between-cloud-grid-and-cluster`, 2017. [Online; accessed 29-June-2017].

[15] R. Li, Q. Zheng, X. Li, and J. Wu, "A novel multi-objective optimization scheme for rebalancing virtual machine placement," in *9th IEEE CLOUD*, pp. 710–717, 2016.

[16] S. Manvi and G. Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424–440, 2014.

[17] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, "Resource provisioning for cloud computing," in *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, pp. 101–111, IBM Corp., 2009.

[18] S. Chaisiri, B. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[19] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[20] S. Chand and M. Wagner, "Evolutionary many-objective optimization: a quick-start guide," *Surveys in Operations Research and Management Science*, vol. 20, no. 2, pp. 35–42, 2015.

[21] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *EMO*, vol. 2632, pp. 376–390, Springer, 2003.

[22] C. Lijun and L. Xiyin, "Modeling server load balance in cloud clusters based on multi-objective particle swarm optimization," *IJGDC*, vol. 8, no. 3, pp. 87–96, 2015.

[23] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[24] C. M. Fonseca, P. J. Fleming, *et al.*, "Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization.," in *Icga*, vol. 93, pp. 416–423, 1993.

[25] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 82–87, Ieee, 1994.

[26] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[27] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[28] R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, 2007.

[29] M. Garza-Fabre, G. Pulido, and C. Coello, "Ranking methods for many-objective optimization," *MICAI 2009: Advances in Artificial Intelligence*, pp. 633–645, 2009.

[30] S. F. Adra and P. J. Fleming, "Diversity management in evolutionary many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 183–195, 2011.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[32] M. Köppen and K. Yoshida, "Substitute distance assignments in nsga-ii for handling many-objective optimization problems," in *Evolutionary multi-criterion optimization*, pp. 727–741, Springer, 2007.

[33] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1, pp. 825–830, IEEE, 2002.

[34] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[35] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 22–31, IEEE, 2000.

[36] Y. Zhu, M. Huang, S. Chen, and Y. Wang, "Energy-efficient topology control in cooperative ad hoc networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1480–1491, 2012.

[37] Q. Zhao, L. Tong, and D. Counsil, "Energy-aware adaptive routing for large-scale ad hoc networks: Protocol and performance analysis," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 9, pp. 1048–1059, 2007.

[38] L. M. Feeney, "An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 6, no. 3, pp. 239–249, 2001.

[39] J.-C. Cano and P. Manzoni, "A performance comparison of energy consumption for mobile ad hoc network routing protocols," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*, pp. 57–64, IEEE, 2000.

[40] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen, "A survey of energy efficient network protocols for wireless networks," *wireless networks*, vol. 7, no. 4, pp. 343–358, 2001.

[41] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre, "A survey on techniques for improving the energy efficiency of large-scale distributed systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 47, 2014.

[42] T. Hirofuchi and A. Lebre, "Adding virtual machine abstractions into simgrid: a first step toward the simulation of infrastructure-as-a-service concerns," in *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pp. 175–180, IEEE, 2013.

[43] F. D. Rossi, M. G. Xavier, Y. J. Monti, and C. A. De Rose, "On the impact of energy-efficient strategies in hpc clusters," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp. 17–21, IEEE, 2015.

[44] E. Outin, J.-E. Dartois, O. Barais, and J.-L. Pazat, "Enhancing cloud energy models for optimizing datacenters efficiency," in *Cloud and Autonomic Computing (ICCAC), 2015 International Conference on*, pp. 93–100, IEEE, 2015.

[45] H. Ismkhan and K. Zamanifar, "Developing improved greedy crossover to solve symmetric traveling salesman problem," *arXiv preprint arXiv:1209.5339*, 2012.

[46] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[47] G. R. Raidl and B. A. Julstrom, "Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem," in *Proceedings of the 2003 ACM symposium on Applied computing*, pp. 747–752, ACM, 2003.

[48] R. K. Ahuja, J. B. Orlin, and A. Tiwari, "A greedy genetic algorithm for the quadratic assignment problem," *Computers & Operations Research*, vol. 27, no. 10, pp. 917–934, 2000.

[49] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[50] A. Bender, "An evaluation of cluster 3.0 as a general tool for principal component analysis," in *Proceedings of the 47th SIGCSE*, pp. 725–725, ACM, 2016.

[51] Sunon, "How to Select the Right Fan or Blower." `http://www.sunon.com/uFiles/file/03_products/07-Technology/004.pdf`, 2016. [Online; accessed 15-August-2016].

[52] Wikipedia, "CFM to Watt Transformation." `http://www.traditionaloven.com/tutorials/power/convert-atm-cfm-atmosphere-to-watts-w.html`, 2016. [Online; accessed 22-February-2016].

[53] M. G. Noll, "Running Hadoop on Ubuntu Linux." `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/`, 2016. [Online; accessed 22-February-2016].

[54] B. Barth, "Determine FLOPS." `https://scicomp.stackexchange.com/questions/11306/how-to-determine-the-amount-of-flops-my-computer-is-capable-of`, 2017. [Online; accessed 29-June-2017].

[55] S. Shah, "FLOPS calculation." `https://www.quora.com/How-do-I-determine-the-amount-of-FLOPs-my-computer-is-capable-of`, 2017. [Online; accessed 29-June-2017].

[56] M. R. Fernandez, "FLOPS related definitions." `http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/2329`, 2017. [Online; accessed 29-June-2017].

[57] A. Rizvi, T. Toha, M. Lunar, M. Adnan, and A. A. Al Islam, "Cooling energy integration in simgrid," in *NSysS*, pp. 132–137, IEEE, 2017.

[58] Purdue University, *MapReduce Benchmark*, July 2017. Available online at `https://engineering.purdue.edu/~puma/pumabenchmarks.htm`. Last accessed July 22, 2017.