POWER-AWARE SCHEDULING FOR SERVER CLUSTERS

POWER-AWARE SCHEDULING FOR SERVER CLUSTERS

By
HADIL AL-DAOUD, B.Eng.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Applied Sciences (M.A.Sc.)

McMaster University

MASTER OF APPLIED SCIENCES (2010) McMaster University
(Software Engineering) Hamilton, Ontario

TITLE: Power-aware Scheduling for Server Clusters

AUTHOR: Hadil Al-Daoud, B.Eng.

SUPERVISOR: Dr. Douglas G. Down

NUMBER OF PAGES: LXXI, 71

Abstract

For the past few years, research in the area of computer clusters has been a hot topic. The main focus has been towards on how to achieve the best performance in such systems. While this problem has been well studied, many of the solutions maximize performance at the expense of increasing the amount of power consumed by the cluster and consequently raising the cost of power usage. Therefore, power management (PM) in such systems has become necessary. Many PM policies are proposed in the literature to achieve this goal for both homogeneous and heterogeneous clusters.

In this work, in the case of homogeneous clusters, we review two applicable policies that have been proposed in the literature for reducing power consumption. We also propose a power saving policy, that uses queueing theory formulas, which attempts to minimize power consumption while satisfying given performance constraints. We evaluate this policy by using simulation and compare it to other applicable policies.

Our main contribution is for heterogeneous clusters. We suggest a task distribution policy in order to reduce power consumption. Our suggested policy requires solving two linear programming problems (LPs). Our simulation experiments show that our proposed policy is successful in terms of achieving a significant power savings in comparison to other distribution policies, especially in the case of highly heterogeneous clusters.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Optimizing performance in computer clusters has been a topic of interest in a number of recent research papers. It is true that research has gone a long way in accomplishing this goal but on the other hand, power consumption has been mostly neglected.

There is almost always a trade-off between performance and energy consumption. Thus, good performance may be attained but at the expense of an undesired level of energy consumption. This is because better performance can be achieved by keeping all machines on all the time in order to handle peak load conditions and improve system responsiveness. Since peak load conditions typically happen infrequently and as a result, most of the time the cluster is underutilized, energy consumption can be reduced significantly just by taking advantage of the times during which the cluster is underutilized.

Reducing energy consumption in computer clusters has become a necessity for many reasons. First of all, for a large cluster which consumes significant amounts of energy, it can be necessary to use expensive cooling equipment. Cooling equipment can consume up to 50% of the total energy consumption in some commercial servers (see Rajamani *et al.* [30]). Also, because of the growing cost of electricity, reducing

energy consumption has become an economic necessity (see Bianchini *et al.* [8]). Furthermore, reducing energy consumption helps the environment since gas emissions during electricity production are reduced (see Chase *et al.* [10]).

In this thesis, we develop scheduling policies which aim to reduce energy consumption in computer clusters. Computer clusters can be homogeneous or heterogeneous. In our study, we consider heterogeneous clusters. Widespread availability of low-cost, high performance computing hardware and the rapid expansion of the Internet and advances in computing networking technology have led to an increasing use of heterogeneous computing (HC) systems (see Kontothanassis and Goddeau [24]). Such systems are constructed by networking various machines with different capabilities and coordinating their use to execute a set of tasks.

Scheduling for such systems is complicated due to several factors. The state of the system dynamically changes and a scheduling policy should adapt its decisions accordingly. Another factor contributing to the complexity of scheduling for clusters is related to the heterogeneous nature of such systems. These systems interconnect a multitude of heterogeneous machines (desktops with various resources: CPU, memory, disk, *etc.*) to perform computationally intensive applications that have diverse computational requirements. Performance may be significantly impacted if information on task and machine heterogeneity is not taken into account by the scheduling policy.

Each machine in a homogeneous cluster consumes power at the same rate. Also, the time it takes for a machine to execute any type of tasks is the same on all machines. In the literature, there has been a lot of work done on power saving in homogeneous clusters. In our work, in the case of homogeneous clusters, we review two power saving policies that were suggested in the literature. Then, we propose a power saving policy that uses basic queueing theory formulas.

In the area of heterogeneous clusters, we suggest a power-aware task distribution policy. Our proposed policy requires solving two linear programming problems in

order to save power while maintaining a particular level of performance. Our experiments shows that our policy provides significant power savings compared to other suggested power saving strategies. Also, we suggest a power saving policy for structured systems that uses the power efficiency of the machines and does not require the knowledge of the arrival rates.

In earlier work ([1] and [2]), several scheduling policies are developed which perform competitively in heterogeneous systems. The policies use the solution to an allocation linear programming problem (LP) which maximizes the system capacity. However, machine power consumption is not considered. In this thesis, we suggest a power-aware scheduling policy (the Power-Aware Linear Programming based Affinity Scheduling policy (LPAS)). The proposed policy also uses the solution to an allocation LP which takes into consideration machine power consumption. Our experiments show that our policy provides significant energy savings in highly heterogeneous systems.

The policy uses the arrival and execution rates to find the maximum capacity. Also, the policy uses information on the power consumption of each machine in order to find an allocation of the machines which results in the maximum energy saving. However, there are cases where obtaining such information is not possible or there is a large degree of uncertainty. In this thesis, we also suggest a power-aware policy for structured systems that only requires knowledge of the ranking of machines with respect to their power efficiencies. Structured systems are a special kind of heterogeneous systems that are common for cluster environments.

This thesis is organized as follows. Section 1.2 gives an overview of the power management policies that are used in the literature. In Section 1.3, the workload model is described and a literature review is given for the case of homogeneous clusters. Section 1.4 also describes the workload model and gives a literature review for the case of heterogeneous clusters. Chapter 2 gives a brief queueing theory background. In Chapter 3, three power saving policies for homogeneous clusters are discussed.

In Chapter 4, we explain and evaluate our proposed Power-Aware task distribution policy for heterogeneous computer clusters. In Chapter 5, we propose a new Power-Aware task distribution policy for structured systems and we evaluate it. Finally, Chapter 6 discusses future work and provides some concluding remarks.

1.2 Background

Finding power management policies for clusters has been the focus of many researchers. Typically, the goal is to meet some performance requirement, while reducing power consumption.

There are two basic power management mechanisms proposed in the literature: Dynamic Voltage Scaling (DVS) and Vary-On/Vary-Off (VOVO). In this section, we give the basic concepts of both mechanisms. Furthermore, we give an overview of the work in the literature that has used these techniques in both homogeneous and heterogeneous clusters. Also, we discuss the relationship of the work in this thesis to the existing literature.

The concept of dynamic voltage scaling is based on the fact that the power consumption P is proportional to the square of the CPU operating voltage V *i.e.* $P \propto V^2$ (Elnozahy *et al.* [13] and Mudge [28]). For example this relation shows that reducing the CPU operating voltage by half will reduce the amount of power consumption by a factor of four. Thus, a considerable amount of power consumption can be saved by decreasing the CPU operating voltage. DVS adjusts the CPU operating voltage V by adapting the maximum CPU operating frequency f according to the intensity of the workload. The relationship between V and f is linear, *i.e.* $V \propto f_{max}$ [13, 28].

By setting the values of the operating voltage and the maximum operating frequency at the lowest values that allow the system to meet performance requirements, the main goal of reducing the cluster's power consumption will be achieved. For in-

stance, if the workload on the system decreases, the maximum operating frequency of the machines can be dynamically decreased and consequently the voltage is decreased.

Machine Vary-On/Vary-Off or the cluster dynamic configuration mechanism uses a subset of available machines to achieve the performance requirements of the system such as meeting tasks' deadlines [13]. Machines that are idling whenever the cluster system is not fully utilized can be either completely turned off or put in low power states. This ensures that power is not wasted when the system is not highly loaded.

1.3 Homogeneous Clusters

In this section, we provide an overview of research in the area of power management for homogeneous clusters. Also, we describe our workload model.

1.3.1 Cluster Workload Model and Assumptions

Suppose we have s homogeneous machines. These machines are homogeneous in terms of both power and speed. Therefore, each machine has the same execution rate μ. The task arrival rate is changing with time and is given by $\lambda(t)$.

The cluster system (see Figure 1.2) has a front-end scheduler that receives periodic messages about the load from the back-end machines. When a task arrives to the system, it is the scheduler's responsibility to assign it to one of the machines. It is assumed that it assigns the tasks to the machines on a First-Come-First-Serve basis. Each machine executes one task at a time and when a machine finishes executing a task, it immediately receives a new task from the scheduler, if there is one available. Furthermore, at regular time intervals (window sizes), the scheduler makes a decision on how many machines to turn on in an effort to minimize power consumption while maintaining sufficiently high performance. In order to decide how many machines to turn on, the scheduler uses a cluster configuration policy. Cluster configuration

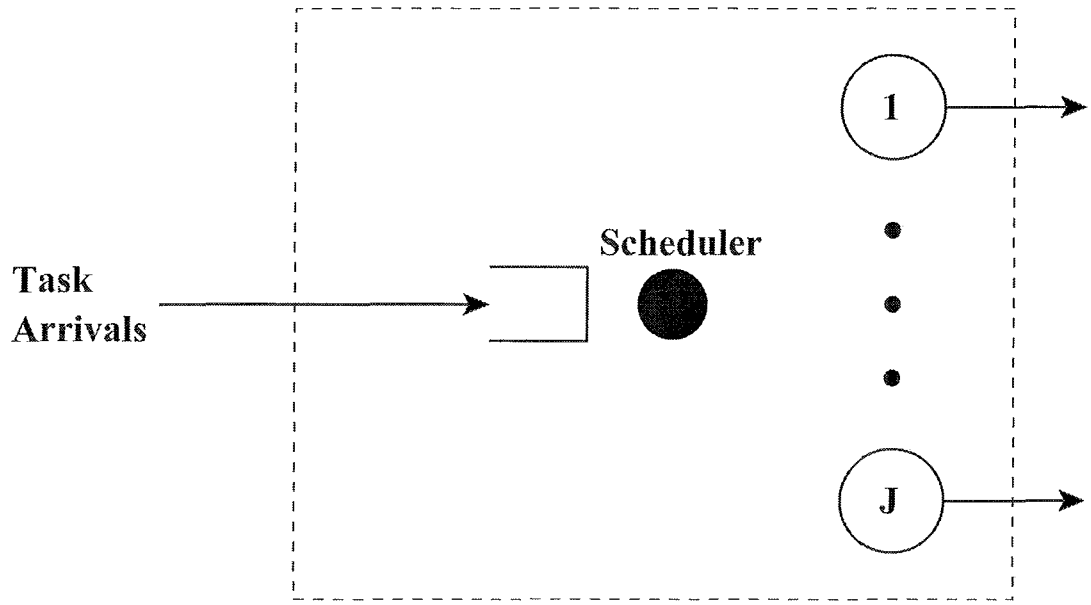

Figure 1.1: Homogeneous Cluster System Model

policies can be static or dynamic.

1.3.2 Related Work

Five policies are proposed in [13]. Independent Voltage Scaling and Coordinated Voltage Scaling are two policies that employ the DVS mechanism in order to reduce the power consumption of each machine. In Independent Voltage Scaling, each machine independently adjusts its CPU operating frequency according to its load. In Coordinated Voltage Scaling, the CPU operating frequency of each machine is set to a desired average. The third policy uses the VOVO mechanism. Two other policies combine the DVS and VOVO mechanisms in order to achieve more energy saving. The first one is a combination of VOVO and Independent Voltage Scaling. The second one is a combination of VOVO and Coordinated Voltage Scaling. The main idea is to adjust the number of operating machines based on a global (target) CPU operating frequency. To clarify, if the global CPU operating frequency increases above a thresh-

old we turn a machine on and if it decreases below this threshold then a machine is turned off.

In Pinheiro *et al.* [29], the authors propose a policy which uses a dynamic cluster configuration mechanism and is based on control theory. Their approach is to dynamically turn machines on and off while keeping performance degradation within acceptable levels. Acceptable performance degradation levels are determined by the system administrator or the user. A machine is turned off if the performance degradation is judged to be acceptable.

In Sharma *et al.* [33], the authors consider a homogeneous cluster with different classes of arriving tasks. The authors show how to lower energy consumption while meeting task deadlines. Both DVS and VOVO mechanisms are used. Meeting task deadlines is achieved by using a technique called synthetic utilization. The CPU operating frequency of each machine is adjusted based on the value of the synthetic utilization. Eventually, if the value of the synthetic utilization is below a certain threshold, the CPU operating frequency of each machine is decreased and vice versa.

Another policy is presented in Elnozahy *et al.* [14]. The authors propose a policy that combines DVS and VOVO mechanisms. In the policy, a subset of the machines are put in a low power state for specified periods of time called batching periods. The response time can be controlled by adjusting the batching period.

1.4 Heterogeneous Clusters

In this section, we describe our workload model. Also, an overview of the research in the area of power management for heterogenous clusters is given.

1.4.1 Cluster Workload Model and Assumptions

In our model for a computer cluster (see Figure 1.2), there is a dedicated front-end scheduler for assigning incoming tasks to the back-end machines. Let the number of

machines in the system be J.

It is assumed that the tasks are classified into I classes. Tasks of class i arrive to the front-end scheduler at rate α_i. Let α be the arrival rate vector, the ith element of α is α_i. The tasks are assumed to be independent and atomic. In the literature, parallel applications whose tasks are independent are sometimes referred to as Bag-of-Tasks applications (BoT) (as in Anglano *et al.* [6]) or parameter-sweep applications (as in Casanova *et al.* [9]). Such applications are becoming predominant for clusters and grids (see Iosup *et al.* [20] and Li and Buyya [26]).

While determining the exact task execution time on a target machine remains a challenge, there exist several techniques that can be used to estimate an expected value for the task execution time (see Rao and Huh [31]). The policies considered in this thesis exploit estimates on mean task execution times rather than exact execution times. Furthermore, in computer clusters and grids, tasks that belong to the same application are typically similar in their resource requirements. For example, some applications are CPU bound while others are I/O bound. In fact, several authors have observed the high dependence of a task's execution time on the application it belongs to and the machine it is running on. They argue for using application profile information to guide resource management (see [24]). We follow the same steps and assume that the tasks are classified into groups (or classes) with identical distributions for the execution times.

Let $\mu_{i,j}$ be the execution rate for tasks of class i at machine j, hence $1/\mu_{i,j}$ is the mean execution time for class i tasks at machine j. We allow $\mu_{i,j} = 0$, which implies machine j is physically incapable of executing class i tasks. Each task class can be executed by at least one machine. Let μ be the execution rate matrix, having (i, j) element $\mu_{i,j}$. Our workload model is similar to the workload model in Al-Azzoni and Down [2].

We note that performance monitoring tools such as NWS [35] and MonALISA [25] can be used to provide dynamic information on the state of the cluster system. Fur-

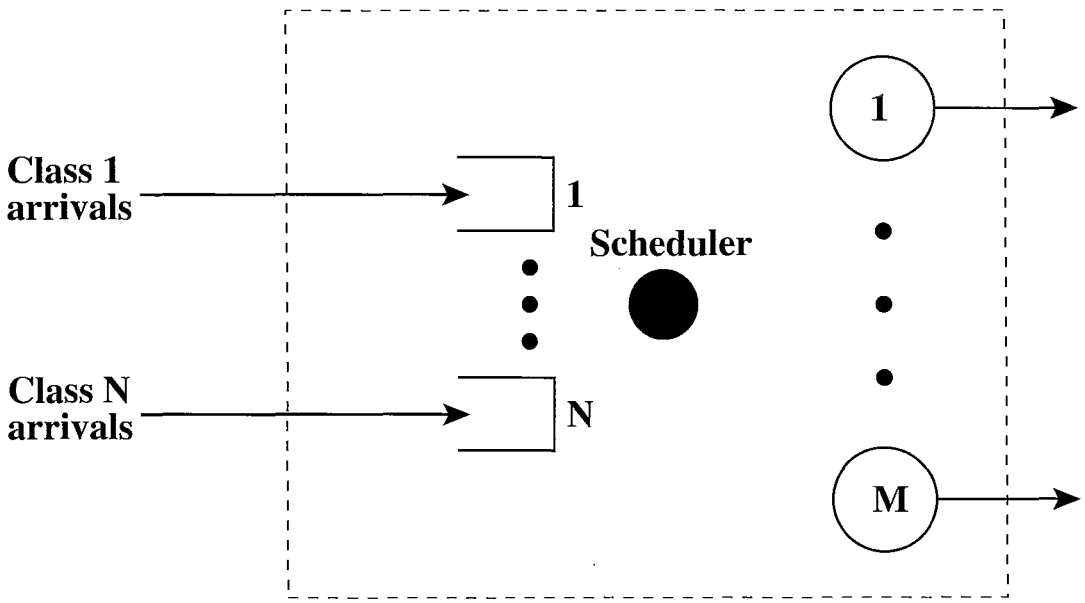

Figure 1.2: Heterogeneous Cluster System Model

thermore, these tools anticipate the future performance behaviour of an application including task arrival and machine execution rates.

At this stage, we introduce the machine power consumption model. We assume that at any point in time a machine can be either busy or in a low power state. Each machine may have different power consumptions when executing different classes of tasks. Let $M_{i,j}$ be the power consumption of machine j when executing a task of class i (it is measured in terms of the energy consumed per time-unit). In addition, we assume that a machine is put into a low power state when it is not executing any task. Let B_j be the power consumption of machine j when it is in a low power state. We assume that $B_j \ll M_{i,j}$ for all i. Our power consumption model is similar to the one considered in Heath *et al.* [19].

1.4.2 Related Work

The energy conservation policy in Heath *et al.* [19] attempts to minimize the total energy consumption-throughput ratio according to predicted load in a heterogeneous cluster. To accomplish this, the authors develop an optimization procedure to find the optimal request distribution policy for the cluster. Analytic models are required to compute the predicted throughputs and total energy consumption. The Power-Aware LPAS policy does not require such analytic models.

In Rusu *et al.* [32], the authors present a policy for reducing energy consumption in heterogeneous clusters while meeting certain requirements on the quality of service (QoS). The proposed policy uses a dynamic cluster configuration mechanism that turns machines on and off according to the system load while ensuring that the QoS requirements are achieved. In addition, they examine the use of the DVS mechanism.

The authors in Guerra *et al.* [17] propose a policy that applies both DVS and VOVO mechanisms in heterogeneous clusters. A linear-programming formalism is employed to find the optimal CPU operating frequency for each machine.

Chapter 2

Preliminaries

In this chapter, we give a brief explanation of some of the basic queueing theory concepts that are used in this thesis.

The exponential distribution is one of the most important and widely used continuous distributions and its probability distribution function is given by:

$$F(x;\lambda) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0 \text{ where } \lambda > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The mean μ and variance σ^2 of an exponentially distributed variable X with rate λ are given by $\mu = 1/\lambda$ and $\sigma^2 = 1/\lambda^2$, respectively.

An important characteristic of the exponential distribution is called the memoryless property. The memoryless property can be stated as follows. Suppose that the inter-arrival times in a queueing system are exponentially distributed with rate λ. This means that knowing the time of the last arrival provides no information on the time to the next arrival, i.e. the time to the next arrival is exponentially distributed with rate λ.

Let the random variable $N(t)$ represent the number of arrivals up to time t. The counting process $N(t), t \geq 0$ is called a Poisson process with rate λ if the inter-arrival times have a common exponential distribution function with rate λ.

Queueing models are used to calculate many steady state performance measures such as the mean task waiting time and the mean number of tasks in the system. A queueing model is represented by Kendall's notation: $A/B/S/K$, where A is the interarrival time distribution, B is the service time distribution, S is the number of servers, and K is the system capacity.

Note that a symbol is used to represent each distribution. For example, the symbol M is used to represent the exponential distribution.

The simplest queueing model is the single server model, known as $M/M/1$. A queueing system is modeled as $M/M/1$ if arrivals follow a Poisson process, execution times are exponentially distributed and there is just a single machine serving arrivals. The mean waiting time is given by the following equation [16]:

$$W = \frac{1}{\mu - \lambda} \tag{2.1}$$

This only holds if the system is stable, which occurs iff:

$$\frac{\lambda}{\mu} < 1. \tag{2.2}$$

Another queueing model is the $M/M/s$ model which generalizes the $M/M/1$ model to a system with multiple servers. A queueing system can be modeled by $M/M/s$ if arrivals follow a Poisson process, execution times are exponentially distributed and there are s machines with a single queue. To compute the mean task waiting time W for an $M/M/s$ model, we use the following formula [16]:

$$W = [\frac{1}{\mu} + \frac{\mu(\frac{\lambda}{\mu})^s}{(s-1)!(s\mu - \lambda)^2}]p_0, \tag{2.3}$$

where p_0 is the probability of having zero tasks in the system and is given by:

$$p_0 = [\sum_{j=1}^{s-1}(\frac{1}{n!})(\frac{\lambda}{\mu})^s + (\frac{1}{s!})(\frac{\lambda}{\mu})^s(\frac{s\mu}{(s\mu - \lambda)})]^{-1}. \tag{2.4}$$

The above formulas assume that the system is stable, *i.e.*,

$$\frac{\lambda}{s\mu} < 1. \tag{2.5}$$

Other queueing systems are analyzed in [16], but these two are sufficient for the purpose of the work in this thesis.

Chapter 3

Homogeneous Clusters

This chapter is organized as follows. In Section 3.1, we give the power model. In Section 3.2, we explain the concept of static cluster configuration. In Section 3.3, we present three dynamic cluster configuration policies. Section 3.4 provides simulation results for these policies. Finally, we give an overview of the related work.

3.1 The On-Off (Power) Model

The state of each server at any time is assumed to be either on or off. Also, the power a machine consumes when it is on is assumed to be P per time unit and zero when it is off. Our power model is similar to the one chosen in [30].

3.2 Static Cluster Configuration Policy

In the static cluster configuration policy, all the machines are turned on all of the time. Therefore, each machine consumes the maximum amount of power even if the load on the whole cluster is very low. To illustrate, suppose we have s machines that are turned on all of the time. Then, the total power consumption rate is sP per time-unit.

3.3 Dynamic Cluster Configuration Policies

Dynamic cluster configuration policies aim to reduce power consumption. Since we assume that the arrival rate is changing, there may be times when the system load is not high. As a result, turning on all the machines will leave some of the machines under-utilized and power is thus wasted. In order to avoid that, the idea is to use the minimum number of machines that is necessary to achieve certain performance requirements.

3.3.1 On/Off Policies

Lien Policy

The policy in Lien *et al.* [27] is suggested to reduce the power consumption for $M/M/1$ workload models. Determining the optimal number of machines is based on maintaining a balance between power and performance. Performance is measured using mean task waiting time W (the time that a task spends in the system from arrival until departure). The idea is to turn on the number of machines that result in the minimum "Mean Task Waiting Time-Power Product". To calculate the mean task waiting time, the $M/M/1$ formula is used. To evaluate their policy, they conduct their experiments on a real server cluster.

In this work, our goal is to generalize the Lien policy for our workload model.

Jennings Policy

The authors in Jennings *et al.* [21] propose a policy for calculating the number of machines needed at specific time instances. The number of machines at time t is determined such that the task's delay probability (using a normal approximation) is no larger than a target value α at all times.

The number of machines that are required as a function of time, $s(t)$, is estimated using the following equation:

$$s(t) = \lceil m(t) + 0.5 + z_\alpha \sqrt{v(t)} \rceil. \tag{3.1}$$

where $\lceil x \rceil$ is the least integer greater than or equal to x, $m(t)$ is the mean function, $v(t)$ is the variance function and z_α satisfies $P(\mathcal{N}(0,1) > z_\alpha) = \alpha$. Note that $\mathcal{N}(m, \sigma^2)$ denotes a Normal random variable with mean m and variance σ^2.

Given that the execution times are exponentially distributed, $m(t)$ is found by solving the following ordinary differential equation (ODE):

$$m'(t) = \lambda(t) - \mu m(t). \tag{3.2}$$

In the case of Poisson arrivals and exponentially distributed execution times, the variance function is given simply by $v(t) = m(t)$.

For the evaluation of this policy, they compare their approximation with the exact numerical solution of the $M_t/M/s_t$ model (the subscript t indicates that both the arrival rate and number of servers vary with time).

3.3.2 The Base Configuration Policy

There is almost always a tradeoff between power and performance. To decrease the mean waiting time, more machines are needed, however, this results in increased power consumption. The base configuration policy applies this concept in order to achieve power saving. It allows the user to choose the level of increase in the average waiting time that can be tolerated and uses this tolerance to reduce the power consumption. The procedure for the dynamic reconfiguration of the cluster using this policy is explained in the following steps.

1) The scheduler calculates the mean waiting time every specified time interval using queueing formulas. We assume that the time intervals (the times at which the

scheduler decides whether to reconfigure the cluster or not) are determined by the user.

2) The performance is measured in our policy by the mean task waiting time. By permitting the mean task waiting time to increase, performance will degrade. To illustrate mathematically, let β be the proportion by which the mean task waiting time is allowed to increase. For example, if we choose $\beta = 0.5$, this will cause the mean task waiting time to increase by 50 percent compared to the mean task waiting time when all the machines are on. The new mean task waiting time is referred to as the target mean waiting time W_{Target}. The user selects the proportion of mean task waiting time degradation (β).

3) The scheduler determines the minimum number of machines to be turned on such that the following condition is satisfied:

$$W_{Target} \leq (1+\beta) W_{AllmachinesOn}. \tag{3.3}$$

This condition guarantees that the target mean task waiting time W_{Target}, does not exceed the mean task waiting time when all machines are on by more than the proportion β.

3.4 Simulation Results

In this section, we present the results of simulating the three proposed power-aware policies. Also, we compare the dynamic configuration policies to the static policy in terms of power consumption and performance.

The arrival process is assumed to be Poisson with rate $\lambda(t)$. Also, every machine has an exponential execution-time distribution with rate μ. Hence, to compute the mean task waiting time W, we use the formula for the $M/M/s$ system (See Chapter 2).

For the Jennings policy, we solve (3.2) to obtain $m(t)$:

$$m(t) = \frac{\lambda}{\mu}(1 - e^{-\mu t}). \tag{3.4}$$

Each experiment is run for 20,000 time-units (seconds) and is repeated 30 times. The cluster consists of 50 machines. We assume that when a machine is turned on, it consumes power at the rate of 100 watts. The execution rate for each machine is 10 tasks per second. Therefore, the maximum arrival rate on the system has to be less than 500 tasks per second. The decision of reconfiguring the cluster is made every 0.3 seconds.

For the static configuration policy, since all the machines are turned on, the total power consumed is just the product of the simulation time, number of machines, and P.

First, we show the tradeoff between power and performance that can be achieved by using the Base Configuration Policy. We take different values for β and then see the impact of changing its value on the power consumption and the average waiting time.

This experiment is conducted for four different values of β (0, 0.01, 0.25, 0.75) for the system under two different arrival rates. The first arrival rate is constant (its value is chosen to be 250). The second one is time-varying (see Table 3.1). The maximum load on the system under $\lambda(t)$ happens when the arrival rate is 470.

Confidence intervals for the mean waiting time were computed at 95% level. We show the average waiting time along with the accuracy of the confidence interval defined as the interval half width divided by the average waiting time. We also show the energy saving (Δ) with respect to a completely turned on system.

Time (t)	$t \leq 100$	$t \leq 150$	$t \leq 250$	$t \leq 300$	$t \leq 400$	$t \leq 450$	$t \leq 600$
Arrival Rate	100	200	450	470	450	200	350
Time (t)	$t \leq 650$	$t \leq 700$	$t \leq 750$	$t \leq 800$	$t \leq 850$	$t \leq 950$	$t \leq 1000$
Arrival Rate	300	250	350	450	300	200	250

Table 3.1: The Arrival Rate

β	Δ	W
0.0	0	$0.10 \pm 0.03\%$
0.01	32.02	$0.10 \pm 0.03\%$
0.25	44.01	$0.12 \pm 0.14\%$
0.75	46.00	$0.13 \pm 0.29\%$

Table 3.2: Simulation Results for the Base configuration policy with constant arrival rate

The results in Table 3.2 and Table 3.3 for both constant and varying arriving rates show that increasing the value of β will degrade the performance *i.e.*, increase the average task waiting time. The amount that the average waiting time will increase depends on the chosen value of β. On the other hand, a reduction in the value of the power consumption will be achieved (because increasing the value of β results in turning on less machines at each decision time). For instance, in the case of a varying arrival rate (Table 3.3), by increasing the value of β from 0.0 to 0.01, the mean waiting time should not exceed 0.101 (This value is calculated using equation (3.3)). Moreover, a 50% power saving is attained. Note that the power saving is calculated compared to the static configuration policy (which is equivalent to the base configuration policy when $\beta = 0$). For the static configuration policy, the power saving is 0% and the average waiting time is $0.10 \pm 0.05\%$.

Based on the results in Table 3.2 and Table 3.3, to get most of the power savings possible with very little loss in performance, we recommend that β should be chosen to be small. For instance, by increasing the value of β from 0.0 to 0.01 in Table 3.2, we save 32.02% at the same value of average waiting time when $\beta = 0$.

Furthermore, we show that a good amount of power saving can be achieved also by using the Lien and Jennings *et al.* policies as opposed to the static configuration

β	Δ	W
0.0	0	$0.100 \pm 0.03\%$
0.01	50.00	$0.100 \pm 0.05\%$
0.25	60.00	$0.102 \pm 0.06\%$
0.75	78.00	$0.180 \pm 0.71\%$

Table 3.3: Simulation Results for the Base configuration policy with changing arrival rate

Policy	Δ	W
Lien	29.03	$0.11 \pm 0.12\%$

Table 3.4: Simulation Results for Lien policy with changing arrival rate

policy. We take the same time-varying arrival rate that is used to evaluate the base configuration policy (see Table 3.1).

The simulation results are shown in Table 3.4 and Table 3.5. The Jennings *et al.* results are done under different values of probability delay α. Also, the results for the Jennings *et al.* policy (Table 3.4) shows that the amount of power saving is increased by increasing the value of α. Also, it can be seen that most of the power savings are achieved after a small increase in α.

3.5 Summary

Power saving for homogeneous clusters can be achieved by applying any of the suggested policies in this chapter. We use two policies which are already proposed in the

α	Δ	W
0.005	16.24	$0.10 \pm 0.10\%$
0.1	27.63	$0.11 \pm 0.10\%$
0.4	33.23	$0.13 \pm 0.36\%$
0.5	35.83	$0.18 \pm 1.17\%$

Table 3.5: Simulation Results for the Jennings *et al.* policy with changing arrival rate

literature and we apply it for our workload model. The first policy is the Lien policy which determines the number of machines that gives the minimum "Mean Task Waiting Time-Power Product". This policy uses queueing theory to calculate the mean waiting time. The second policy is proposed by Jennings *et al.* In this policy, the delay probability of each task is kept below a target value. The power saving can be increased or decreased according to the chosen target value.

In this chapter, we also propose another basic power saving policy that can be used in homogeneous clusters. This policy, which we call the base configuration policy, considers the tradeoff between power reduction and performance which gives the opportunity to emphasize power savings over performance or vice versa. The average waiting time is directly controllable by the user and can be kept below a target value. The determination of the minimum number of machines is based on the selected value for the percentage of the average waiting time.

One should be aware of the following when applying the previous policies. First, it is easier for system administrators to express service level agreements using the average task waiting time rather than a task's delay probability. Both Lien and the base configuration policies require the former parameter as the performance target, while Jennings *et al.* policy requires the latter. Second, in the Lien and the base configuration policies, one needs to use queueing theory to estimate the mean waiting times. In some cases, closed-form solutions do not exist and thus one must use other approaches, which may be expensive. Finally, for all policies, one should set the window size. Although not discussed in the thesis, the window size should be chosen based on the dynamics of the system. For example, if the arrival rate is changing rapidly, one may choose very small window sizes.

Some aspects are not considered in our approach and need further study. For example, we have not considered the delay time and cost obtained when a machine changes from one power state to another. Also, we have not considered issues such as choosing the window size.

Chapter 4

Heterogeneous Clusters

Designing a power-efficient heterogeneous server cluster is studied in this chapter. More specifically, we propose a power-aware task distribution strategy for the heterogeneous clusters. As previously mentioned, the purpose of finding such a strategy is to schedule the incoming tasks in a way that the total power consumed by the cluster is minimized.

The organization of this chapter is as follows. In Section 4.1, we define our power model. In Section 4.2, we describe our proposed power-aware task distribution policy. Section 4.3 demonstrates the experimental results. The content of this chapter and the following chapter appears in [3] and Al-Daoud *et al.* [4].

4.1 The On-Off Model

At any point in time, a machine can be either busy or in a low power state. Each machine consumes a different amount of power per time-unit when executing each type of tasks.

Assume that $M_{i,j}$ represents the power consumption of machine j when executing a task of type i, where $i = 1, ..., I$ and $j = 1, ..., J$. In addition, we assume that a

machine is put into a low power state (*i.e.*, a deep sleep state) when it is not executing any tasks or idling. Let B_j be the power consumption of machine j when it is in the deep sleep state. We assume that $B_j \ll M_{i,j}$ for all i *i.e.*, the power consumed by a machine in a deep sleep state is much lower than the amount of power consumed when it is on.

Let P_j be the average power consumed per unit time by a machine j; therefore, the total power consumed by all machines can be expressed as:

$$(4.1)\ \sum_{j=1}^{J} P_j = \sum_{j=1}^{J}[(\sum_{i=1}^{I} \delta_{i,j} M_{i,j}) + (1 - \sum_{i=1}^{I} \delta_{i,j}) B_j]$$

where $\delta_{i,j}$ is the proportion of time that machine j is busy executing tasks of type i. Our power model is similar to the one given in Heath *et al.* [19].

4.2 Current Policies

A scheduling policy that is applicable to our workload model is the classical First-Come-First-Serve (FCFS) policy. FCFS is used in major schedulers (such as Domingues *et al.* [11] and Kondo *et al.* [23]). An advantage of FCFS is that it does not require any dynamic information on the state of the system. However, FCFS only performs well in systems with limited task heterogeneity and under moderate system loads. As the application tasks become more heterogeneous and the load increases, performance degrades rapidly (see Al-Azzoni and Down [1]). Furthermore, FCFS ignores machine power consumption and thus may result in serious energy waste.

Another scheduling policy is the Pick-the-Most-Efficient (PME) policy. The policy uses a greedy approach for assigning tasks to machines. It is defined as follows. When

a machine j becomes available, it is assigned a class i task where the power efficiency of machine j on class i is the maximum amongst those classes with at least one task waiting. The power efficiency of a machine j on class i tasks is defined as $\mu_{i,j}/M_{i,j}$. The PME policy only requires dynamic information on the machine execution rates and power consumption. In particular, it does not take into account information on the task arrival rates.

4.3 The Power-Aware LPAS Policy

4.3.1 Introduction

In the LPAS policy (see [2]), the power-efficiency of the cluster is not considered. In this work, we suggest a scheduling policy that takes power consumption into account and performs well. This policy is called: Power-Aware Linear Programming Based Affinity Scheduling (Power-Aware LPAS). This policy takes advantage of low load periods by carefully choosing (using an LP solution) which machines to idle and be put into a low-power state. A machine is activated only when there is a need *e.g.*, executing an arriving task.

In this section, we give a basis for our Power-Aware task distribution policy with an example that should help to clarify how it works. Also, we illustrate its performance using simulation results.

4.3.2 Main Concept of Power-Aware LPAS policy

The Power-Aware LPAS policy requires solving two allocation linear programming (LP) problems. The first LP does not take power consumption into account. It is the same LP that is used in the other LPAS-related policies (see [1] and [2]). This LP is solved for the purpose of obtaining the maximum capacity of the system λ^*. This value is then used in the second LP.

In the first LP, the decision variables are λ and $\theta_{i,j}$ for $i = 1, \ldots, I$, $j = 1, \ldots, J$. The variables $\theta_{i,j}$ are to be interpreted as the proportional allocation of machine j to class i.

$$\max \ \lambda$$

(4.2a)

$$\text{s.t.} \ \sum_{j=1}^{J} \theta_{i,j}\mu_{i,j} \geq \lambda\alpha_i, \quad \text{for all } i = 1, \ldots, I,$$

(4.2b)

$$\sum_{i=1}^{I} \theta_{i,j} \leq 1, \quad \text{for all } j = 1, \ldots, J,$$

(4.2c)

$$\theta_{i,j} \geq 0, \quad \text{for all } i = 1, \ldots, I, \text{ and } j = 1, \ldots, J.$$

The left-hand side of (4.2a) represents the total execution capacity assigned to class i by all machines in the system. The right-hand side represents the arrival rate of tasks that belong to class i scaled by a factor of λ. Thus, (4.2a) enforces that the total capacity allocated for a class should be at least as large as the scaled arrival rate for that class. This constraint is needed to have a stable system. The constraint (4.2b) prevents overallocating a machine and (4.2c) states that negative allocations are not allowed.

Let λ^* and $\{\theta^*_{i,j}\}$, $i = 1, \ldots, I$, $j = 1, \ldots, J$, be an optimal solution to LP (4.2). The LP always has a solution, since no lower bound constraint is put on λ. However, the physical meaning of λ^* requires that its value be at least one (as explained below). In this case, $1/\lambda^*$ is interpreted as the long-run utilization of the busiest machine.

The value λ^* can also be interpreted as the maximum capacity of the system. We define the maximum capacity as follows. Consider a system with given values for α_i $(i = 1, \ldots, I)$ and λ^*. If $\lambda^* \leq 1$, then the system is unstable. Thus, the system will be overloaded and tasks queue indefinitely. If, however, $\lambda^* > 1$, then the system can be

stabilized even if each arrival rate is increased by a factor less than or equal to λ^* (*i.e.*, the same system with arrival rates $\alpha'_i \leq \lambda^* \alpha_i$, for all $i = 1, \ldots, I$, can be stabilized). In this case, the values $\{\theta^*_{i,j}\}$, $i = 1, \ldots, I$, $j = 1, \ldots, J$, can be interpreted as the long-run fraction of time that machine j should spend on class i in order to stabilize the system under maximum capacity conditions.

The second LP considers the power consumption of the machines. The decision variables are $\delta_{i,j}$ for $i = 1, \ldots, I$, $j = 1, \ldots, J$.

$$\min \sum_{j=1}^{J} [(\sum_{i=1}^{I} \delta_{i,j} M_{i,j}) + (1 - \sum_{i=1}^{I} \delta_{i,j}) B_j]$$

(4.3a)

$$\text{s.t.} \sum_{j=1}^{J} \delta_{i,j} \mu_{i,j} \geq c\alpha_i, \quad \text{for all } i = 1, \ldots, I,$$

(4.3b)

$$\sum_{i=1}^{I} \delta_{i,j} \leq 1, \quad \text{for all } j = 1, \ldots, J,$$

(4.3c)

$$\delta_{i,j} \geq 0, \quad \text{for all } i = 1, \ldots, I, \text{ and } j = 1, \ldots, J.$$

The constraint (4.3a) enforces that the total execution capacity allocated for a class should be at least as large as the arrival rate for that class scaled by a factor c. The optimal solution for this LP is given in the form of a matrix δ^* where the (i, j) entry is $\delta^*_{i,j}$. The matrix δ^* specifies an allocation of machines to tasks such that the energy consumption is minimized while still meeting capacity c.

The Power-Aware LPAS policy considers the trade-off between energy consumption and performance. Let c represent the target capacity of the system. Assuming that $\lambda^* > 1$, the value of c can range from 1 to the maximum capacity of the system, *i.e.*, $1 \leq c \leq \lambda^*$. In this case, LP (4.3) always has a solution, since θ^* already satisfies

the constraints (4.3a)-(4.3c). Choosing for c values closer to 1 may cause performance to degrade while achieving increased energy saving. If c is very close to 1, then only the minimum capacity is utilized and this results in severe performance degradation (or even system instability). Thus, we avoid the use of such values for c. On the other hand, the closer c to the maximum capacity λ^*, the better the performance, at the expense of increased energy consumption.

In order to achieve the allocations $\delta^*_{i,j}$, we use the following mechanism. Suppose that machine j requests a task at time t. Let $\delta_{i,j}(t)$ be the proportion of time that machine j has been executing class i tasks up to time t. The scheduler assigns the machine to a class i task such that $\delta^*_{i,j} > 0$ and $\delta^*_{i,j}$ - $\delta_{i,j}(t)$ is the maximum. If all of the values of $\delta^*_{i,j}$ - $\delta_{i,j}(t)$ are negative, machine j is put in a low power state until $\frac{L_j(t)}{t} = 1 - \sum_{i=1}^{I} \delta^*_{i,j}$, where $L_j(t)$ is the total time machine j has been in a low power state up to time t.

Consider a system with two machines and two classes of tasks ($I = 2$, $J = 2$). The arrival and execution rates are as follows:

$$\alpha = \begin{bmatrix} 1 & 1.5 \end{bmatrix} \text{ and } \mu = \begin{bmatrix} 9 & 5 \\ 2 & 1 \end{bmatrix}.$$

Furthermore, assume that

$$B = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix} \text{ and } M = \begin{bmatrix} 1 & 20 \\ 1 & 20 \end{bmatrix}.$$

Thus, when executing a task, power consumption of machine 2 is 20 times that of machine 1. Both machines have the same power consumption in the low power state.

Solving LP (4.2) gives $\lambda^* = 1.7647$ and

$$\theta^* = \begin{bmatrix} 0 & 0.3529 \\ 1 & 0.6471 \end{bmatrix}.$$

First, set $c = \lambda^*$. Solving LP (4.3) gives $\delta^* = \theta^*$. The resulting δ^* achieves the maximum system capacity (see [1]), however it ignores power consumption of the

machines. Machine 2 is assigned tasks for execution although it is very inefficient power-wise.

In the second case we set $c = 1$. Solving LP (4.3) gives

$$\delta^* = \begin{bmatrix} 0.1111 & 0 \\ 0.7500 & 0 \end{bmatrix}.$$

Note that in this case machine 2 is put in a low power state. The allocation δ^* results in the maximum energy saving while meeting the minimum required capacity.

4.3.3 Simulation Results

Overview

We use simulation to compare the performance of the scheduling policies. In Section 4.3.4, we simulate artificial systems with different heterogeneities to show the impact of heterogeneity on performance. Then, in Section 4.3.5, we show the results of simulating a realistic cluster system.

The task arrivals are modeled by independent Poisson processes, each with rate α_i, $i = 1, \ldots, I$. The execution times are exponentially distributed with rates $\mu_{i,j}$, where $1/\mu_{i,j}$ represents the mean execution time of a task of class i at machine j, $i = 1, \ldots, I$, $j = 1, \ldots, J$.

There are several performance metrics that can be used. We use the long-run average task completion time W, as a metric for performance comparison. A task completion time is defined as the time elapsing between the submission of the task and the completion of its execution. Another metric we also show is the energy saving (Δ) with respect to FCFS.

Each simulation experiment models a particular system, characterized by the values of I, J, B_j, $M_{i,j}$, α_i, and $\mu_{i,j}$, $i = 1, \ldots, I$, $j = 1, \ldots, J$. Each experiment is executed for 20,000 time-units and repeated 30 times. For every case, we give W and Δ. For W, we also give the accuracy of the confidence interval defined as the ratio of

Policy	c	Δ	W
Power-Aware LPAS	$\lambda^* = 1.7068$	38.21%	$0.165 \pm 0.24\%$
Power-Aware LPAS	midpoint=1.3534	45.63%	$0.265 \pm 1.97\%$
PME	-	13.20%	$0.261 \pm 0.22\%$
FCFS	-	0%	$2.842 \pm 14.08\%$

Table 4.1: Simulation Results for Experiment 1

Policy	c	Δ	W
Power-Aware LPAS	$\lambda^* = 1.4582$	22.38%	$0.308 \pm 0.45\%$
Power-Aware LPAS	midpoint=1.2291	54.14%	$0.335 \pm 1.92\%$
PME	-	4.41%	$0.207 \pm 0.23\%$
FCFS	-	0%	$0.207 \pm 0.25\%$

Table 4.2: Simulation Results for Experiment 2

the half width of the interval over the mean value (all statistics are at 95% confidence level).

4.3.4 Task and Machine Heterogeneity

There are different kinds of system heterogeneity. Machine heterogeneity refers to the average variation along the rows of μ, and similarly task heterogeneity refers to the average variation along the columns (see Armstrong [7]). In the first experiment, we simulate a system with high task heterogeneity and high machine heterogeneity. In the second experiment, we simulate a system with high machine heterogeneity and low task heterogeneity. In both experiments, machine power consumptions are completely heterogeneous.

Experiment 1

Consider a system with 3 classes and 6 machines ($I = 3$, $J = 6$). The system is chosen to be both highly machine and task heterogeneous. The arrival and execution rates for this system are given by $\alpha = [9.75\ 8.5\ 9.5]$ and

$$\mu = \begin{bmatrix} 4.5 & 2 & 9.5 & 6.2 & 10.25 & 2.25 \\ 6.2 & 4.5 & 6 & 2 & 4.2 & 5.9 \\ 9.5 & 6.5 & 4 & 10 & 5.9 & 2.25 \end{bmatrix}, \text{ respectively.}$$

The following define machine power consumption:

$$B = \begin{bmatrix} 3.5 & 3 & 4 & 4 & 3.5 & 3 \end{bmatrix}$$

and

$$M = \begin{bmatrix} 66 & 73 & 84 & 103 & 93 & 75 \\ 50 & 65 & 79 & 71 & 82 & 63 \\ 105 & 80 & 96 & 85 & 95 & 70 \end{bmatrix}.$$

Solving LP (4.2) gives $\lambda^* = 1.7068$. Table 4.1 shows the simulation results for the experiment. The table gives simulation results for the Power-Aware LPAS policy under two different values of c: $c = \lambda^*$ and $c = \frac{1+\lambda^*}{2}$.

The results show that significant energy saving can be achieved by using the Power-Aware LPAS policy. When c is set to the midpoint (*i.e.*, $\frac{1+\lambda^*}{2}$), the Power-Aware LPAS policy results in an energy saving that is almost 2.5 times that of PME while achieving the same performance.

Experiment 2

In this experiment, we consider a system with high machine heterogeneity and low task heterogeneity. The system has 6 machines and 3 classes ($I = 3$, $J = 6$). The arrival and execution rates are respectively given by $\alpha = [8.75\ 8.5\ 9]$ and

$$\mu = \begin{bmatrix} 2.2 & 7 & 10.25 & 1 & 5.7 & 12 \\ 1.95 & 7.05 & 9.78 & 0.95 & 5.65 & 11.85 \\ 2 & 7.25 & 10.02 & 0.98 & 5.75 & 11.8 \end{bmatrix}.$$

The following define machine power consumption:

$$B = \begin{bmatrix} 3.5 & 3 & 4 & 4 & 3.5 & 3 \end{bmatrix}$$

and

$$M = \begin{bmatrix} 128.4 & 193.1 & 155.6 & 105.5 & 125.4 & 116.1 \\ 135.1 & 230.15 & 203.4 & 94.2 & 250.6 & 85.5 \\ 84.15 & 62.3 & 81.1 & 96.9 & 71.3 & 215.09 \end{bmatrix}.$$

Solving LP (4.2) gives $\lambda^* = 1.4582$. Table 4.2 shows the simulation results for the experiment. The table gives simulation results for the Power-Aware LPAS policy under two different values of c: $c = \lambda^*$ and $c = \frac{1+\lambda^*}{2}$.

The results show that using the Power-Aware LPAS policy results in significant energy saving compared to both FCFS and PME but at the expense of an increased average waiting time. Note that the system has low task heterogeneity. In such systems, previous work has demonstrated that LPAS-related policies may not perform as well as other competing policies (see [1] and [2]).

Experiment 3

Consider a system with high machine heterogeneity and high task heterogeneity. This system also has $M = 6$ and $N = 3$. The arrival rates vary every 100 time-units and are given by the following vectors:
$\alpha_1 = \begin{bmatrix} 9.75 & 8.5 & 9.5 \end{bmatrix}$, $\alpha_2 = \begin{bmatrix} 7.75 & 7.5 & 7.5 \end{bmatrix}$, $\alpha_3 = \begin{bmatrix} 9.75 & 8.5 & 9.5 \end{bmatrix}$and $\alpha_4 = \begin{bmatrix} 7.75 & 7.5 & 7.5 \end{bmatrix}$.

The results in Table 4.3 show that significant energy saving can be also achieved by using the Power-Aware LPAS policy in the case of varying arrival rates.

Policy	c	Δ	W
Power-Aware LPAS	λ^*	30.94	$0.16 \pm 0.33\%$
Power-Aware LPAS	midpoint	34.86	$0.20 \pm 0.02\%$
Pick-Most-Efficient	-	0	$0.25 \pm 0.32\%$

Table 4.3: Simulation Results for Experiment 3

4.3.5 Realistic Architectures

In this section, we simulate a system which models a real computer cluster [24] (for details, see He [18]) to compare the scheduling policies. The system is a medium size system with 5 task classes and 30 machines. The machines are partitioned into 6 groups, machines within a group are identical. Thus, if two machines are in the same group, then they have the same execution rates. Groups T, U, V, W, X, and Y, consist of 2 machines, 6 machines, 7 machines, 7 machines, 4 machines, and 4 machines, respectively. The execution rates are shown in Table 4.4. The arrival rate vector is $\alpha = [204.10\ 68.87\ 77.63\ 5.01\ 10.43]$. For such a system, $\lambda^* = 2.4242$.

We consider two cases. In the first case, machine power consumptions are completely heterogeneous. The machine power consumption matrix M is shown in Table 4.5. $M_{1,\ldots,10}$ is a sub-matrix of M which shows the power consumption for machines $1, \ldots, 10$ (the sub-matrices $M_{11,\ldots,20}$ and $M_{21,\ldots,30}$ are defined analogously). Machines in Group T are 1 and 2, machines in Group U are $3, \ldots, 8$, *etc.*

	Group					
Task	T	U	V	W	X	Y
1	16.7	24.8	24.2	29	25.6	48.3
2	30.4	48.3	77.7	83.6	135.9	144.9
3	18.9	24.2	48.3	45.8	72.5	72.5
4	3	3	7.6	7.6	8.3	8.7
5	1	1.1	3	2.9	3	3

Table 4.4: The Execution Rates for the System in Section 4.3.5

$$M_{1,\ldots,10} = \begin{bmatrix} 53.2 & 70.1 & 67.2 & 45.3 & 48.8 & 78.5 & 120.0 & 163.1 & 77.3 & 85.0 \\ 82.6 & 200.7 & 148.8 & 68.8 & 92.9 & 97.9 & 87.4 & 67.0 & 78.3 & 94.4 \\ 216.3 & 79.2 & 94.3 & 86.5 & 218.6 & 87.8 & 96.4 & 136.9 & 200.3 & 136.1 \\ 97.2 & 87.4 & 136.4 & 154.5 & 156.1 & 176.2 & 137.3 & 183.9 & 149.6 & 230.6 \\ 120.0 & 123.0 & 65.0 & 78.0 & 94.4 & 132.1 & 79.3 & 88.8 & 99.5 & 100.2 \end{bmatrix}$$

$$M_{11,\ldots,20} = \begin{bmatrix} 93.3 & 64.1 & 82.6 & 72.9 & 59.1 & 69.1 & 59.3 & 75.4 & 88.0 & 130.6 \\ 90.6 & 69.7 & 84.4 & 73.3 & 120.2 & 102.1 & 160.7 & 210.3 & 93.7 & 190.8 \\ 164.2 & 89.3 & 95.5 & 189.6 & 129.6 & 87.5 & 74.8 & 98.0 & 94.9 & 129.0 \\ 94.8 & 86.9 & 94.1 & 78.4 & 76.6 & 98.0 & 75.3 & 120.2 & 134.4 & 160.2 \\ 90.4 & 65.0 & 73.0 & 97.9 & 179.0 & 213.0 & 169.8 & 61.2 & 123.0 & 145.5 \end{bmatrix}$$

$$M_{21,\ldots,30} = \begin{bmatrix} 116.7 & 69.3 & 150.4 & 144.5 & 78.0 & 96.0 & 73.5 & 180.7 & 211.0 & 130.0 \\ 211.9 & 94.2 & 89.3 & 67.5 & 87.6 & 73.7 & 133.8 & 128.0 & 123.0 & 221.6 \\ 137.0 & 129.2 & 234.1 & 176.2 & 146.3 & 197.4 & 136.6 & 79.4 & 83.6 & 76.1 \\ 96.9 & 130.6 & 143.4 & 176.1 & 109.3 & 79.1 & 69.6 & 78.9 & 143.3 & 165.5 \\ 135.3 & 123.6 & 89.5 & 68.8 & 85.9 & 90.2 & 143.9 & 156.7 & 189.3 & 67.5 \end{bmatrix}$$

Table 4.5: The Machine Power Consumption Matrix for the System in Section 4.3.5 - The Heterogeneous Case

The second case represents more homogeneous per-cluster power consumption. We assume that the power consumption for a machine is just a multiple of its execution rate. Thus, the faster the machine, the more energy it consumes. Furthermore, the multiplicative factor is different amongst the groups. This represents realistic systems in which the machines in a cluster are homogeneous in terms of their power consumption while the clusters differ in their power efficiency. The multiplicative factors are 6, 4, 7, 5.5, 5, and 6, for groups T, U, V, W, X, and Y, respectively.

In both cases, the power consumption in the low power state is 2 for machines in Group T, 3 for machines in Groups U, V, and X, 3.5 for machines in Group W, and 4 for machines in Group Y. For the Power-Aware LPAS policy, we give simulation results corresponding to five different values of c (1.1500, 1.3561, 1.7121, 2.0682, and 2.4242).

Figures 4.1 and 4.2 show the simulation results under both cases. The figures show that the Power-Aware LPAS policy performs competitively while reducing energy consumption. The improvements are more significant in systems with higher degrees of heterogeneity. Also, when the parameter c is set to values closer to λ^*, better performance results. In this case, since the system being simulated is not highly loaded (41.25%), performance improvement is not that significant. However, if the load increases, performance improvement becomes much more significant.

The Power-Aware LPAS policy results in reduced energy consumption, ranging from 25% to 50% in the heterogeneous case and from 0.5% to 5.5% in the more homogeneous case. We note that the energy saving is not linear with respect to decreasing values of c (the same observation holds for performance with respect to increasing values of c). Furthermore, when c is set to the midpoint (*i.e.*, $\frac{1+\lambda^*}{2} = 1.7160$), the Power-Aware LPAS policy results in a reasonable compromise between performance improvement and energy saving. An administrator of a cluster can adjust the value of c to tailor to the organization's specific need. For example, one can reduce c just below the midpoint if energy consumption is more of a concern than

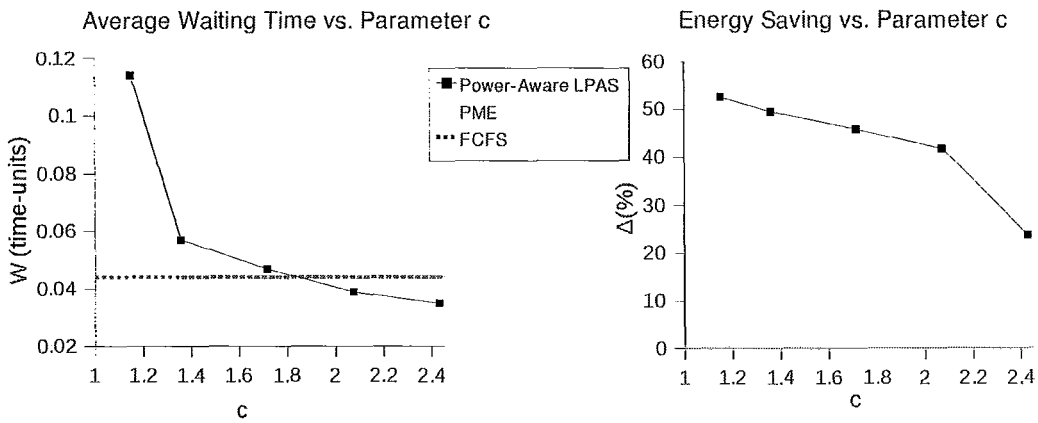

Figure 4.1: Simulation results for the system in Section 4.3.5 - The heterogeneous case

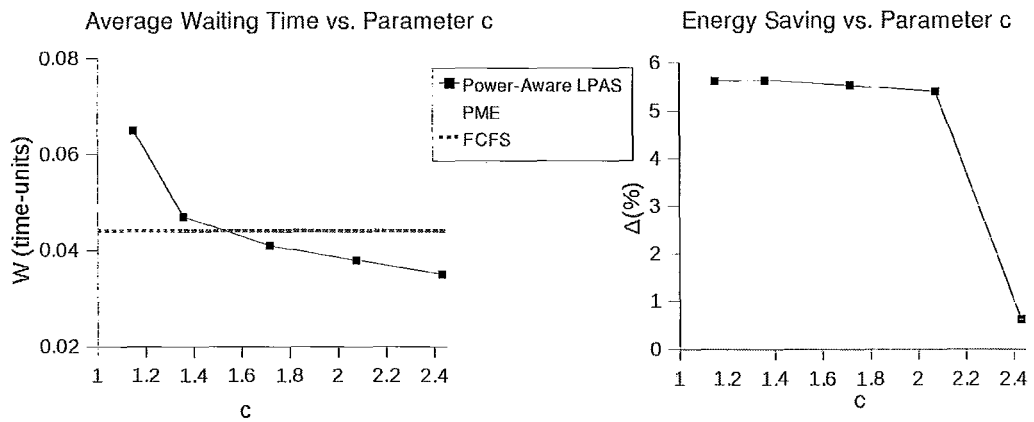

Figure 4.2: Simulation results for the system in Section 4.3.5 - The more homogeneous case

performance.

4.4 Comparison to Dynamic Voltage Scaling

The Power-Aware LPAS policy employs a dynamic cluster configuration mechanism in which a machine is put in a low power state when it is not executing a task. Another

power management mechanism is the Dynamic Voltage Scaling mechanism in which a machine can have different CPU operating frequencies. At any time, a machine's low power state and busy power consumption depend on the current machine operating frequency. Both mechanisms were discussed in Section 1.2.

Consider the following Dynamic Voltage Scaling policy. The utilization of each machine U_j is computed periodically and then the CPU operating frequency for machine j is set to the closest value higher than $U_j f_{max}$, where f_{max} is the maximum CPU operating frequency of machine j. The policy is used in Govil *et al.* [15] and Rusu *et al.* [32].

In this section, we use the same system specified in Section 4.3.5 which models a real computer cluster. However, to simulate the system using Dynamic Voltage Scaling, we need to scale the execution rates as well as each machine's low power state and busy power consumption. The scaling is based on the power consumption parameters of a real machine used in the experiments of [32] and which are reproduced in the following table:

Frequency (MHz)	1000	1800	2000	2200	2400
Idle (Watts)	70	74.5	78.5	83.5	89.5
Busy (Watts)	80.5	92.5	103.5	119.5	140.5

In our simulation experiments, we assume that all of the machines have the same parameters as in the table above. Let f'_j be the current CPU operating frequency for machine j. Then, the scaling is done as follows:

- $\mu'_{i,j} = \frac{f'_j}{2400}\mu_{i,j}$, where μ is the execution rate matrix from Section 4.3.5. Note that 2400 is the maximum CPU operating frequency of the modelled machine..
- $M'_{i,j} = \frac{\text{machine } j \text{ busy power consumption corresponding to } f'_j}{140.5} M_{i,j}$, where M is the machine busy power consumption matrix from Section 4.3.5. Note that 140.5 is the busy power consumption corresponding to the maximum CPU operating frequency of the modelled machine.

- $B'_j = \frac{\text{machine } j \text{ idle power consumption corresponding to } f'_j}{89.5} B_j$, where B is the machine low power state power consumption vector from Section 4.3.5. Note that 89.5 is the idle power consumption corresponding to the maximum CPU operating frequency of the modelled machine.

We assume that the front-end scheduler uses FCFS. Each machine sets its CPU operating frequency every 0.0001 time-units. This high frequency allows the machines to quickly adapt their CPU operating frequencies while meeting the load. We give the simulation results for the two cases in Section 4.3.5. For the heterogeneous machine power consumption case, using Dynamic Voltage Scaling results in an average task waiting time that is 93.78% higher than that of FCFS with $\Delta = -75.64\%$. For the more homogeneous machine power consumption case, using Dynamic Voltage Scaling results in an average task waiting time that is 92.69% higher than that of FCFS with $\Delta = -81.79\%$. These results indicate that Dynamic Voltage Scaling does not perform well in systems with high task heterogeneity. Both the average task waiting time and the energy consumption increase. This is because the original DVS based policies implicitly assume homogeneity. It is possible that they could be adapted to the heterogeneous case.

4.5 Summary

The main contribution in this chapter is the suggestion of the Power-Aware LPAS policy for heterogeneous clusters. The Power-Aware LPAS policy requires solving two allocation LPs in order to find the the best task distribution that results in the lowest power consumed by the system. First, the maximum capacity is computed by solving the first LP. Then, a value for the system capacity between the maximum value and 1 is chosen. Then, another LP that takes the power consumption into consideration is solved. By solving the second LP, the best allocation matrix for the chosen value of the system capacity is found. Our simulation experiments show that by using the

Power-Aware LPAS policy, our goal of minimizing the power consumption is achieved.

Chapter 5

Power-Aware LPAS for Structured Systems

5.1 Introduction

In the previous chapter, a good amount of energy saving can be achieved by using the Power-Aware LPAS policy. However, the Power-Aware LPAS policy requires knowledge of both task arrival and machine execution rates. In this chapter, we introduce another energy saving policy for heterogeneous structured systems which does not require knowledge of the task arrival rates.

5.2 Structured Systems

5.2.1 Introduction

The execution rate matrix of a structured system is given by a combination of two components: a component that is completely dependent on the task (the inherent task difficulty) and another component that is completely dependent on the machine (the machine efficiency). Such systems appear to be reasonable models for computer

cluster environments. Thus, the execution rate of machine j on a class i task is given by $\mu_{i,j} = \gamma_j \mu_i$, $i = 1, \ldots, I$, $j = 1, \ldots, J$.

The busy power consumption matrix is also structured such that each machine's power consumption is equal to a factor multiplied by its speed. So, the power consumption of machine j while executing a class i task can be given by $M_{i,j} = \beta_j \mu_{i,j}$, $i = 1, \ldots, I$, $j = 1, \ldots, J$, where the factor $1/\beta_j$ is the power efficiency of machine j.

Suppose we have a system with $M = 7$ machines and $N = 4$ tasks. To formulate the execution rate matrix, we choose: $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 5$, $\mu_4 = 3$. Then, we select values for γ_j. For example, suppose that: $\gamma_1 = 1$, $\gamma_2 = 3$, $\gamma_3 = 4$, $\gamma_4 = 0.2$, $\gamma_5 = 6$, $\gamma_6 = 5$, $\gamma_7 = 10$. As a result of multiplying μ_i by γ_j, the execution rate matrix is given by:

$$\mu = \begin{bmatrix} 1 & 3 & 4 & 0.2 & 6 & 5 & 10 \\ 2 & 6 & 8 & 0.4 & 12 & 10 & 20 \\ 5 & 15 & 20 & 1 & 30 & 25 & 50 \\ 3 & 9 & 12 & 0.6 & 18 & 15 & 30 \end{bmatrix}.$$

To form the power matrix corresponding to the execution rate matrix, we multiply each column of the execution rate matrix by β_j. For instance, we select the following values for β_j: $\beta_1 = 3.1$, $\beta_2 = 11.7$, $\beta_3 = 8.2$, $\beta_4 = 6.5$, $\beta_5 = 13.6$, $\beta_6 = 17.4$, $\beta_7 = 1.3$.

Thus, the busy power consumption matrix is given by:

$$M = \begin{bmatrix} 3.1 & 35.1 & 32.8 & 1.3 & 81.6 & 87 & 13 \\ 6.2 & 70.2 & 65.6 & 2.6 & 163.2 & 174 & 26 \\ 15.5 & 175.5 & 164 & 6.5 & 408 & 435 & 65 \\ 9.3 & 105.3 & 98.4 & 3.9 & 244.8 & 261 & 39 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 3 & 3 & 0.5 & 3 & 3 & 3 \end{bmatrix}.$$

Table 5.1 shows simulation results for this system under the constant arrival rate vector: $\alpha = \begin{bmatrix} 6.25 & 6 & 6.25 & 6 \end{bmatrix}$. Also, we simulate the system under arrival rates that change every 100 time-units. At every arrival rate change event, the arrival rate vector assumes the values in one of the following vectors: $\alpha_1 = \begin{bmatrix} 6.25 & 6 & 6.25 & 6 \end{bmatrix}$, $\alpha_2 = \begin{bmatrix} 3.75 & 3.6 & 3.75 & 3.6 \end{bmatrix}$, $\alpha_3 = \begin{bmatrix} 6.25 & 6 & 6.25 & 6 \end{bmatrix}$ and $\alpha_4 = \begin{bmatrix} 3.75 & 3.6 & 3.75 & 3.6 \end{bmatrix}$. Table (5.2) shows the simulation results.

Policy	c	Δ	W
Power-Aware LPAS	$\lambda^* = 2.3360$	40.93	$0.167 \pm 0.13\%$
Power-Aware LPAS	midpoint= 1.6680	57.13	$0.20 \pm 0.32\%$
Pick-Most-Efficient	-	0.009	$0.164 \pm 0.10\%$
FCFC	-	0	$0.163 \pm 0.11\%$

Table 5.1: Simulation Results

Policy	c	Δ	W
Power-Aware LPAS	λ^*	41.59	$0.172 \pm 0.27\%$
Power-Aware LPAS	midpoint	57.77	$0.199 \pm 5.47\%$
Pick-Most-Efficient	-	0	$0.175 \pm 0.41\%$
FCFS	-	0	$0.175 \pm 0.43\%$

Table 5.2: Simulation Results

5.2.2 Observation

For structured systems, the machines should be put in a low power state in increasing order of β_j when the load on the system is reduced and employed in decreasing order of β_j when the load on the system is increased.

Consider a system that has the same characteristics of the system described in Section 5.2.1. Table 5.3 shows $\sum_{i=1}^{I} \delta_{i,j}^*$ for each machine j (*i.e.*, the load of the machine) at different values of the system load ($\frac{1}{\lambda^*}$) assuming c is set to the midpoint ($\frac{1+\lambda^*}{2}$).

For instance, at an arrival rate $\alpha = \begin{bmatrix} 13.75 & 13.2 & 13.75 & 13.2 \end{bmatrix}$, the load on the system is 0.9418. At the midpoint ($c = 1.0309$), the resulting allocation matrix is:

$$\delta^* = \begin{bmatrix} 0.0000 & 1.0000 & 0.2563 & 0.0000 & 1.0000 & 0.8300 & 0.0000 \\ 1.0000 & 0.0000 & 0.7437 & 1.0000 & 0.0000 & 0.0000 & 0.2629 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2835 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.4536 \end{bmatrix}.$$

Note that the sum of each column of the δ^* matrix is 1, hence all machines are

allocated all of the time.

Now, by decreasing the arrival rates to $\alpha = \begin{bmatrix} 12.5 & 12 & 12.5 & 12 \end{bmatrix}$, the load on the system will be reduced to 0.856. At the midpoint ($c = 1.084$), the resulting allocation matrix is:

$$\delta^* = \begin{bmatrix} 0.0000 & 1.0000 & 0.4125 & 0.0000 & 1.0000 & 0.5800 & 0.0000 \\ 1.0000 & 0.0000 & 0.5875 & 1.0000 & 0.0000 & 0.0000 & 0.2954 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2710 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.4336 \end{bmatrix}.$$

Notice that machine 6 (which has the largest β) is the first machine for which $\sum_{i=1}^{I} \delta^*_{i,j}$ is zero and thus it is the first machine to be put in the low power state. If we decrease the load further and compute $\sum_{i=1}^{I} \delta^*_{i,j}$ for each machine j, the machines are put in a low power state in decreasing order of β_j: machines 5, 2, 3, 4, 1, then 7 (or equivalently, increasing order of the power efficiency *i.e.*, $\frac{1}{\beta_j}$). In fact, we can show that putting machines in a low power state in order of their power efficiencies as the load decreases and vice versa characterizes a particular subset of optimal solutions to LP (4.3).

Lemma 1 *For a structured system where $B_j = 0$ for $j = 1, \ldots, J$, if there are two machines j_1 and j_2 such that $\beta_{j_1} > \beta_{j_2}$, we can not have: $\sum_i \delta^*_{i,j_1} > 0$ and $\sum_i \delta^*_{i,j_2} = 0$ in an optimal solution for LP (4.3).*

Proof

We prove the lemma by contradiction as follows.

Consider a structured system with J machines and I classes. Assume that for two machines j_1 and j_2 we have $\beta_{j_1} > \beta_{j_2}$.

Suppose that in an optimal solution, we have $\sum_{i=1}^{I} \delta^*_{i,j_1} > 0$ and $\sum_{i=1}^{I} \delta^*_{i,j_2} = 0$.

load	M1	M2	M3	M4	M5	M6	M7
0.9418	1	1	1	1	1	0.83	1
0.856	1	1	1	1	1	0.58	1
0.7705	1	1	1	1	1	0.3301	1
0.6849	1	1	1	1	1	0.08	1
0.5993	1	1	1	1	0.8584	0	1
0.5137	1	1	1	1	0.6499	0	1
0.428	1	1	1	1	0.4417	0	1
0.3425	1	1	1	1	0.2333	0	1
0.2568	1	1	1	1	0.025	0	1
0.1712	1	0.6333	1	1	0	0	1
0.08556	1	0.2167	1	1	0	0	1
0.000856 (c=500)	1	0	0.325	1	0	0	1
0.000856 (c=430)	0.75	0	0	0	0	0	1
0.000856 (c= 195)	0	0	0	0	0	0	1

Table 5.3: Load on each machine for different loads on the system

The value of the objective function at this optimal solution is then given by:

$$(5.1)\quad \sum_{j\neq j_2}\sum_{i=1}^{I}\mu_i\gamma_j\beta_j\delta^*_{i,j}.$$

Consider another solution $\overline{\delta^*}$ constructed as follows. Let $\overline{\delta^*}$ be identical to δ^* except for the columns corresponding to machines j_1 and j_2. Let $\overline{\delta^*}_{i,j_1} = \gamma_{j_1}/(\gamma_{j_1}+\gamma_{j_2})\delta^*_{i,j_1}$ and $\overline{\delta^*}_{i,j_2} = \gamma_{j_1}/(\gamma_{j_1}+\gamma_{j_2})\delta^*_{i,j_1}$ for $i = 1, ..., I$.

First, we show that the constructed solution is a feasible solution, *i.e.*, it satisfies (4.3a)-(4.3c).

To show that the constructed solution satisfies (4.3a), note that:

$$
\begin{aligned}
&\sum_{j=1}^{J} \overline{\delta^*}_{i,j}\mu_{i,j} \\
&= \sum_{j\neq j_1,j_2} \delta^*_{i,j}\mu_{i,j} + \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}\mu_{i,j_1} + \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}\mu_{i,j_2} \\
&= \sum_{j\neq j_1,j_2} \delta^*_{i,j}\mu_{i,j} + \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}[\mu_{i,j_1}+\mu_{i,j_2}] \\
&= \sum_{j\neq j_1,j_2} \delta^*_{i,j}\mu_{i,j} + \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}\mu_i[\gamma_{j_1}+\gamma_{j_2}] \\
&= \sum_{j\neq j_1,j_2} \delta^*_{i,j}\mu_{i,j} + \gamma_{j_1}\delta^*_{i,j_1}\mu_i \\
&= \sum_{j\neq j_2} \delta^*_{i,j}\mu_{i,j} \\
&\geq c\alpha_i, \text{ for } i = 1,\ldots,I.
\end{aligned}
$$

To show that (4.3b) is satisfied, note that for j_1:

$$
\begin{aligned}
&\sum_{i=1}^{I} \overline{\delta^*}_{i,j_1}\mu_{i,j_1} \\
&= \sum_{i=1}^{I} \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}\mu_{i,j_1} \\
&= \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}} \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_{i,j_1} \\
&\leq 1 \text{ since } \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}} \leq 1 \text{ and } \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_{i,j_1} \leq 1.
\end{aligned}
$$

For j_2, one can show that $\sum_{i=1}^{I} \overline{\delta^*}_{i,j_2}\mu_{i,j_2} \leq 1$ as follows:

$$\begin{aligned}
&\sum_{i=1}^{I} \overline{\delta^*}_{i,j_2}\mu_{i,j_2} \\
&= \sum_{i=1}^{I} \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}\mu_{i,j_2} \\
&= \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}} \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_{i,j_2} \\
&= \frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}} \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_i\gamma_{j_2} \\
&= \frac{\gamma_{j_2}}{\gamma_{j_1}+\gamma_{j_2}} \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_i\gamma_{j_1} \\
&= \frac{\gamma_{j_2}}{\gamma_{j_1}+\gamma_{j_2}} \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_{i,j_1} \\
&\leq 1 \text{ since } \frac{\gamma_{j_2}}{\gamma_{j_1}+\gamma_{j_2}} \leq 1 \text{ and } \sum_{i=1}^{I} \delta^*_{i,j_1}\mu_{i,j_1} \leq 1.
\end{aligned}$$

For all other machines j,

$$\sum_{i=1}^{I} \overline{\delta^*}_{i,j}\mu_{i,j} = \sum_{i=1}^{I} \delta^*_{i,j}\mu_{i,j} \leq 1.$$

Hence, constraint (4.3b) holds.

Finally, note that $\overline{\delta^*}_{i,j} \geq 0$ for all j and hence (4.3c) is satisfied.

The new objective function is given by:

$$(5.2)\quad \sum_{j\neq j_1,j_2} \sum_{i=1}^{I} \mu_i\gamma_j\beta_j\overline{\delta^*}_{i,j} + \sum_{i=1}^{I} \mu_i\gamma_{j_1}\beta_{j_1}\overline{\delta^*}_{i,j_1} + \sum_{i=1}^{I} \mu_i\gamma_{j_2}\beta_{j_2}\overline{\delta^*}_{i,j_2}.$$

By substituting $\overline{\delta^*}_{i,j}, i = 1,\ldots,I, j = 1,\ldots,J$, in (5.2), the resulting objective function value is:

$$\sum_{j\neq j_1,j_2}\sum_{i=1}^{I}\mu_i\gamma_j\beta_j\overline{\delta^*_{i,j}}+\sum_{i=1}^{I}\mu_i\gamma_{j_1}\beta_{j_1}\frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}$$
$$+\sum_{i=1}^{I}\mu_i\gamma_{j_2}\beta_{j_2}\frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}\delta^*_{i,j_1}$$
$$=\sum_{j\neq j_1,j_2}\sum_{i=1}^{I}\mu_i\gamma_j\beta_j\overline{\delta^*_{i,j}}$$
$$+(\gamma_{j_1}\beta_{j_1}\frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}}+\gamma_{j_2}\beta_{j_2}\frac{\gamma_{j_1}}{\gamma_{j_1}+\gamma_{j_2}})\sum_{i=1}^{I}\mu_i\delta^*_{i,j_1}$$
$$=\sum_{j\neq j_1,j_2}\sum_{i=1}^{I}\mu_i\gamma_j\beta_j\delta^*_{i,j}+\beta_{j_1}\gamma_{j_1}\frac{\gamma_{j_1}+(\beta_{j_2}/\beta_{j_1})\gamma_{j_2}}{\gamma_{j_1}+\gamma_{j_2}}\sum_{i=1}^{I}\mu_i\delta^*_{i,j_1}.$$

Since $\beta_{j_2}/\beta_{j_1}<1$, it follows that

$$\frac{\gamma_{j_1}+(\beta_{j_2}/\beta_{j_1})\gamma_{j_2}}{(\gamma_{j_1}+\gamma_{j_2})}<1.$$

Thus, the corresponding value of the new objective function (5.2) is smaller than that of (5.1). Hence, the constructed solution is an optimal solution contradicting our original assumption and the proof is complete.

The following theorem is a direct implication of the lemma.

Theorem 1 *As the value of c is decreased, the Power-Aware LPAS turns machines off in descending order of β_j. Conversely, as the value of c is increased, the Power-Aware LPAS turns machines on in ascending order of β_j.*

5.2.3 A New Scheduling Policy

Introduction

As a direct implication, we propose a Power-Aware policy that turns on and off machines in the order of β_j and we call this policy the ordered β policy. The ordered β policy uses the following parameters: the window size (WS), the target waiting time (W_{Target}) and the threshold (T). The window size determines the decision points.

Policy	c	Δ	W
the ordered β	-	40.38	$0.177 \pm 0.33\%$
Power-Aware LPAS	$\lambda^* = 2.3360$	40.93	$0.167 \pm 0.13\%$
Pick-Most-Efficient	-	0.009	$0.164 \pm 0.10\%$
FCFS	-	0	$0.163 \pm 0.11\%$

Table 5.4: Simulation Results

After every WS time units, the scheduler computes the average waiting time for the tasks that are executed during the interval. The parameters W_{Target} and T determine when a new machine should be added to those being employed or a working machine should be put in a low power state. A new machine is added to those employed when the average waiting time is above $(1 - T)W_{Target}$ and an additional machine is put in a low power state when the average waiting time is below $(1 - 2T)W_{Target}$, where $0 < T < 1$. The machines to be added to those employed or to be put in a low power state are chosen according to the ordering of β_j, as explained earlier.

The ordered β policy only requires knowledge of the ranking of the machines in terms of their power efficiencies. It does not require the task arrival or execution rates of the machines, nor their power consumptions. This is extremely useful in systems where obtaining such information is difficult or there is a large degree of uncertainty.

Simulation Results

Consider a system that has the same execution rate and power consumption matrices as the system mentioned in Section 5.2.1. We simulate this system under both constant and varying arrival rates.

Under the constant arrival rate $\alpha = \begin{bmatrix} 6.25 & 6 & 6.25 & 6 \end{bmatrix}$, the results of the ordered β policy compared to other policies are given in Table 5.4. This table gives information about the average task waiting time W and the improvement in energy saving with respect to the FCFS policy. Note that these results are taken under the following values for the parameters: $WS = 25$, $W_{Target} = 0.2$ and $T = 0.1$.

Policy	c	Δ	W
the ordered β	-	38.27	$0.165 \pm 0.36\%$
Power-Aware LPAS	λ^*	41.59	$0.172 \pm 0.27\%$
Pick-Most-Efficient	-	0	$0.175 \pm 0.41\%$
FCFS	-	0	$0.175 \pm 0.43\%$

Table 5.5: Simulation Results

For the same varying arrival rate used in Section 5.2.1, the results of the ordered β policy compared to other policies are given in Table 5.5. These results are taken under the following values for the parameters: $WS = 250$, $W_{Target} = 0.2, 0.15, 0.2, 0.15$ and $T = 0.25$. The target average waiting time W_{target} is taken to be equal to the average waiting time W in the Power-Aware LPAS at $c = \lambda^*$. The results are taken compared to the FCFS policy.

The results in Table 5.5 show significant energy saving achieved by the ordered β policy. Also, the achieved amount of energy saving is comparable to that of the Power-Aware LPAS policy which requires knowledge of the arrival and execution rates as well as the machine power consumptions.

5.2.4 Structured Systems Approximation

Introduction

In this section, we propose a technique for finding machine efficiencies β_j in systems that are not exactly structured. We give two examples on how to do that.

Method for Finding the Power Efficiency

Consider a variation on the structured system above (see Section 5.2.1) such that $\mu_{i,j} = \gamma_j \mu_i (1 + \varepsilon^a_{i,j})$ and $M_{i,j} = \beta_j \mu_{i,j} (1 + \varepsilon^b_{i,j})$, $i = 1, \ldots, I$, $j = 1, \ldots, J$. The inaccuracy levels $\varepsilon^a_{i,j}$ and $\varepsilon^b_{i,j}$ are sampled from the uniform distribution on $[-0.5, 0.5]$.

To find the machine power efficiencies (β_j) in a system that is not exactly struc-

tured, the following procedure can be used. First, an approximation to the closest structured matrix to μ is found by applying singular value decomposition (see Strang [34]). Singular value decomposition is a factorization of an $m \times n$ matrix in the form USV^T where U is an m-by-m unitary matrix, S is an m-by-n diagonal matrix with non-negative real numbers on the diagonal, and V^T (an n-by-n unitary matrix) is the conjugate transpose of V.

Based on the Eckart-Young theorem [12], a matrix A can be approximated by a rank r matrix $\tilde{A}$, where $\tilde{A} = U\tilde{S}V^T$, in which $\tilde{S}$ is the same matrix as S except that it contains only the r largest singular values (the other singular values are replaced by zeros). By using this theorem, the execution rate matrix μ can be approximated by a rank 1 matrix $\tilde{\mu}$. The resulting matrix $\tilde{\mu}$ is the closest structured matrix to μ.

The machine power efficiencies can then be found using linear regression (see Autar and Kalu [22]). β_j is set to the slope of the straight line that best fits the data points $(\tilde{\mu}_{i,j}, M_{i,j}), i = 1, \ldots, I$. Several linear regression methods exist. The simplest method is the ordinary least squares method which minimizes the sum of squared residuals [22].

Examples and Simulation Results

Consider the structured system discussed in Section (5.2.1) where:

$$\mu = \begin{bmatrix} 10 & 1 & 0.2 & 4 & 3 & 6 & 5 \\ 20 & 2 & 0.4 & 8 & 6 & 12 & 10 \\ 50 & 5 & 1 & 20 & 15 & 30 & 25 \\ 30 & 3 & 0.6 & 12 & 9 & 18 & 15 \end{bmatrix} \text{ and M} = \begin{bmatrix} 13 & 3.1 & 1.3 & 32.8 & 35.1 & 81.6 & 87 \\ 26 & 6.2 & 2.6 & 65.6 & 70.2 & 163.2 & 174 \\ 65 & 15.5 & 6.5 & 164 & 175.5 & 408 & 435 \\ 39 & 9.3 & 3.9 & 98.4 & 105.3 & 244.8 & 261 \end{bmatrix}.$$

Each simulation experiment is repeated 30 times. Each table of results shows the average waiting time and the percentage of energy saving for the ordered β policy compared to the FCFS policy at three different inaccuracy levels (5%, 10% and 50%). For the average waiting time, we also give the accuracy of the confidence interval. We generate 30 matrices at every inaccuracy level for $\mu_{i,j}$.

We give a couple of examples. In each one, we generate a execution rate matrix μ

and a corresponding power matrix under a certain inaccuracy level. Then, we show how to find the values of β_j for the non-exact structured system. Then, we show the energy saving that can be achieved by applying the ordered β policy and compare it to both Power-Aware LPAS and the FCFS policies in terms of performance and energy saving.

In the first example, we generate a system with 50 percent inaccuracy using the structured system:

$$\mu = \begin{bmatrix} 14.6049 & 0.5169 & 0.2881 & 5.6078 & 1.7936 & 6.1267 & 2.9235 \\ 25.9080 & 2.7378 & 0.2383 & 11.0422 & 3.0360 & 8.4241 & 9.7264 \\ 57.2116 & 7.1698 & 0.5984 & 21.4180 & 15.1137 & 34.7831 & 34.8229 \\ 37.0726 & 3.5678 & 0.7856 & 8.8849 & 11.3716 & 22.8371 & 15.6354 \end{bmatrix},$$

$$\mathrm{M} = \begin{bmatrix} 18.9864 & 1.6025 & 1.8726 & 45.9843 & 20.9853 & 83.3236 & 50.8690 \\ 33.6805 & 8.4872 & 1.5490 & 90.5458 & 35.5216 & 114.5683 & 169.2392 \\ 74.3750 & 22.2265 & 3.8898 & 175.6273 & 176.8308 & 473.0501 & 605.9181 \\ 48.1944 & 11.0600 & 5.1064 & 72.8562 & 133.0481 & 310.5845 & 272.0560 \end{bmatrix},$$

$$\mathrm{B} = \begin{bmatrix} 3 & 1 & 0.5 & 3 & 3 & 3 & 3 \end{bmatrix},$$

$$\alpha = \begin{bmatrix} 5.5 & 5 & 5.5 & 5 \end{bmatrix}.$$

By applying the singular value decomposition, we get the following matrices:

$$\mathrm{U} = \begin{bmatrix} -0.1653 & -0.4784 & 0.2661 & -0.8203 \\ -0.3046 & -0.8063 & -0.1926 & 0.4691 \\ -0.8036 & 0.3147 & -0.4738 & -0.1754 \\ -0.4839 & 0.1483 & 0.8171 & 0.2761 \end{bmatrix},$$

$$\tilde{S} = \begin{bmatrix} 99.6771 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Policy	c	Δ	W
the ordered β	-	77.78	$0.229 \pm 0.78\%$
Power-Aware LPAS	midpoint=1.47925	74.07	$0.217 \pm 0.60\%$
FCFS	-	0	$0.182 \pm 0.14\%$

Table 5.6: Results of Experiment 1.

$$V = \begin{bmatrix} -0.7446 & -0.4919 & 0.2950 & 0.2585 & -0.1513 & -0.0089 & 0.1638 \\ -0.0843 & 0.0372 & -0.1236 & 0.4057 & -0.0974 & 0.4846 & -0.7532 \\ -0.0098 & -0.0028 & 0.0552 & -0.0087 & -0.3308 & -0.8056 & -0.4882 \\ -0.2588 & -0.3968 & -0.4994 & -0.4992 & 0.4512 & -0.0479 & -0.2667 \\ -0.1893 & 0.3528 & 0.2868 & 0.3049 & 0.7727 & -0.2401 & -0.0986 \end{bmatrix}.$$

The closest rank 1 matrix to μ is:

$$\begin{bmatrix} 12.2685 & 1.3890 & 0.1615 & 4.2642 & 3.1190 & 7.0388 & 6.4457 \\ 22.6073 & 2.5595 & 0.2975 & 7.8576 & 5.7475 & 12.9705 & 11.8775 \\ 59.6428 & 6.7525 & 0.7850 & 20.7300 & 15.1630 & 34.2189 & 31.3353 \\ 35.9148 & 4.0661 & 0.4727 & 12.4829 & 9.1306 & 20.6055 & 18.8690 \end{bmatrix}.$$

Applying linear regression, we obtain the following values for β: β_1= 1.3, β_2 = 3.1, β_3 = 6.5, β_4 = 8.2, β_5= 11.7, β_6= 13.6 and β_7= 17.4.

Then, we apply the ordered β policy and compare it to the Power-Aware LPAS policy and the FCFS policy. The results of the simulation are shown in Table 5.6. Note that the ordered β policy has the following parameters: the window size = 100, the desired waiting time = 0.3 and the threshold = 0.1. The results show that the energy saving achieved by the ordered β policy is comparable to that achieved by the Power-Aware LPAS policy.

In the second example, we generate a system with 10 percent inaccuracy using the following structured system:

$$\mu = \begin{bmatrix} 10.9210 & 0.9034 & 0.2176 & 4.3216 & 2.7587 & 6.0253 & 4.5847 \\ 21.1816 & 2.1476 & 0.3677 & 8.6084 & 5.4072 & 11.2848 & 9.9453 \\ 51.4423 & 5.4340 & 0.9197 & 20.2836 & 15.0227 & 30.9566 & 26.9646 \\ 31.4145 & 3.1136 & 0.6371 & 11.3770 & 9.4743 & 18.9674 & 15.1271 \end{bmatrix},$$

$$M = \begin{bmatrix} 14.1973 & 2.8005 & 1.4145 & 35.4369 & 32.2771 & 81.9447 & 79.7738 \\ 27.5361 & 6.6574 & 2.3898 & 70.5892 & 63.2643 & 153.4737 & 173.0478 \\ 66.8750 & 16.8453 & 5.9780 & 166.3255 & 175.7662 & 421.0100 & 469.1836 \\ 40.8389 & 9.6520 & 4.1413 & 93.2912 & 110.8496 & 257.9569 & 263.2112 \end{bmatrix},$$

$$B = \begin{bmatrix} 3 & 1 & 0.5 & 3 & 3 & 3 & 3 \end{bmatrix},$$

$$\alpha = \begin{bmatrix} 5.5 & 5 & 5.5 & 5 \end{bmatrix}.$$

By applying singular value decomposition, we get the following matrices:

$$U = \begin{bmatrix} -0.1614 & -0.4777 & 0.2645 & -0.8221 \\ -0.3167 & -0.8030 & -0.1919 & 0.4670 \\ -0.8013 & 0.3228 & -0.4699 & -0.1814 \\ -0.4813 & 0.1511 & 0.8200 & 0.2705 \end{bmatrix},$$

$$\tilde{S} = \begin{bmatrix} 88.2353 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$V = \begin{bmatrix} -0.7345 & -0.4894 & 0.2891 & 0.2697 & -0.1807 & 0.0044 & 0.1791 \\ -0.0757 & 0.0387 & -0.1233 & 0.4087 & -0.1316 & 0.4559 & -0.7651 \\ -0.0135 & -0.0034 & 0.0548 & -0.0061 & -0.3237 & -0.8304 & -0.4500 \\ -0.2851 & -0.3987 & -0.5054 & -0.4763 & 0.4481 & -0.0507 & -0.2722 \\ -0.2126 & 0.3496 & 0.2843 & 0.3301 & 0.7629 & -0.2265 & -0.0970 \\ -0.4361 & 0.5175 & 0.3071 & -0.5968 & -0.1799 & 0.1853 & -0.1577 \\ -0.3715 & 0.4583 & -0.6839 & 0.2611 & -0.1733 & -0.1204 & 0.2677 \end{bmatrix}.$$

The closest rank 1 matrix to μ is:

$$\begin{bmatrix} 10.4601 & 1.0781 & 0.1923 & 4.0602 & 3.0277 & 6.2106 & 5.2906 \\ 20.5250 & 2.1154 & 0.3772 & 7.9669 & 5.9409 & 12.1864 & 10.3812 \\ 51.9313 & 5.3522 & 0.9545 & 20.1574 & 15.0314 & 30.8336 & 26.2661 \\ 31.1925 & 3.2148 & 0.5733 & 12.1075 & 9.0286 & 18.5201 & 15.7767 \end{bmatrix}.$$

Applying a linear regression, we obtain the following values for β: $\beta_1 = 1.3$, $\beta_2 = 3.1$, $\beta_3 = 6.5$, $\beta_4 = 8.2$, $\beta_5 = 11.7$, $\beta_6 = 13.6$ and $\beta_7 = 17.4$.

Policy	c	Δ	W
the ordered β	midpoint=1.2612	66.94	$0.387 \pm 2.26\%$
Power-Aware LPAS	-	73.76	$0.372 \pm 1.23\%$
FCFS	-	0	$0.172 \pm 0.11\%$

Table 5.7: Results of Experiment 2.

Policy	c	Δ	W
the ordered β	-	9.88	$0.049 \pm 0.07\%$
Power-Aware LPAS	1.3580	5.63	$0.047 \pm 0.17\%$
Pick-the-most-efficient	-	0	$0.044 \pm 0.07\%$
FCFS	-	0	$0.044 \pm 0.07\%$

Table 5.8: Simulation Results

In this example, the ordered β policy is simulated under the following parameters: the window size = 50, the desired waiting time = 0.35 and the threshold = 0.1. The results presented in Table 5.7 shows that energy saving achieved by the ordered β policy is comparable to that achieved by the Power-Aware LPAS policy.

The second case of the realistic system (the more homogeneous case) (see Section 4.3.5) is also a non-exact structured system. By doing the singular value decomposition, we get the same values for β_j as in the model description. Table 5.8 shows the results of the ordered β policy under the following values for the parameters: $WS = 0.001$, $W_{target} = 0.04$ and $T = 0.25$. Note that the target average waiting time W_{target} in the ordered β policy is taken to be equal to the average waiting time W in the Power-Aware LPAS policy at $c = 1.3580$.

5.3 Summary

In this chapter, we suggested an energy saving policy for structured systems that does not require knowledge of the arrival or execution rates. This policy only requires knowledge of the relative machine power efficiencies. We also described a method for

finding the machine power efficiencies in such cases where the system is not exactly structured. The results of the experiments show that the energy saving that can be achieved by the ordered β policy is comparable to that achieved by applying the Power-Aware LPAS (which requires knowledge of the arrival rates).

Chapter 6

Conclusion

This thesis addressed the important problem of power saving in both homogeneous and heterogeneous clusters. Our main contribution is the proposition of a new power-aware scheduling policy for heterogeneous clusters. This policy seeks to provide significant energy saving by solving two allocation LPs. The first LP is solved to find the maximum system capacity, while the second is solved to find an optimal allocation of machines to minimize the energy consumption. Our simulation results demonstrate that significant energy saving can be achieved compared to other policies. For structured systems, we also suggest a policy which only requires the machine power efficiencies and results in competitive energy saving and performance. As future work, we plan to implement the proposed policies on a real heterogeneous cluster in order to validate the simulation results.

One limitation of using linear programming in scheduling is scalability. Solving LPs for a large system may incur significant delay which can impact the performance of the Power-Aware LPAS policy. Also, in highly dynamic systems, the parameters may change very frequently causing performance degradation due to the overhead and delays of solving the LPs. In such cases, it is recommended to use other scheduling policies including the ordered β policy for structured systems.

Another aspect of the Power-Aware LPAS policy which requires further study is

robustness. Robustness can be defined as the degree to which a system can function correctly in the presence of parameter values different from those assumed (Ali *et al.* [5]). As observed in [2], the solution to the allocation LP is inherently robust and thus we expect the Power-Aware LPAS policy to be robust.

Bibliography

[1] I. Al-Azzoni and D. G. Down. Dynamic scheduling for heterogeneous Desktop Grids. In *Proceedings of the 9th International Conference on Grid Computing*, pages 136–143, 2008.

[2] I. Al-Azzoni and D. G. Down. Linear programming-based affinity scheduling of independent tasks on heterogeneous computing systems. 19(12):1671–1682, 2008.

[3] H. Al-Daoud, I. Al-Azzoni, and D. G. Down. Power-Aware Linear Programming Based Scheduling for heterogeneous computer clusters, submitted for publication, 2010.

[4] H. Al-Daoud, I. Al-Azzoni, and D. G. Down. Power-Aware Linear Programming Based Scheduling for heterogeneous computer clusters. In *Proceedings of the Workshop in Progress in Green Computing Workshop*, 2010.

[5] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 15(7):630–641, 2004.

[6] C. Anglano, J. Brevik, M. Canonico, D. Nurmi, and R. Wolski. Fault-aware scheduling for Bag-of-Tasks applications on Desktop Grids. In *Proceedings of the 7th International Conference on Grid Computing*, pages 56–63, 2006.

[7] R. Armstrong. Investigation of effect of different run-time distributions on SmartNet performance. Master's thesis, Naval Postgraduate School, 1997.

[8] R. Bianchini and R. Rajamony. Power and energy management for server systems. *Computer*, 37(11):68–74, 2004.

[9] H. Casanova, D. Zagorodnov, F. Berman, and A. Legrand. Heuristics for scheduling parameter sweep applications in grid environments. In *Proceedings of the 9th Heterogeneous Computing Workshop*, pages 349–363, 2000.

[10] J. S. Chase and R. P. Doyle. Balance of power: Energy management for server clusters. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS)*, 2001.

[11] P. Domingues, P. Marques, and L. Silva. DGSchedSim: A trace-driven simulator to evaluate scheduling algorithms for desktop grid environments. In *Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 83–90, 2006.

[12] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[13] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Proceedings of the Second International Workshop of Power-Aware Computer Systems*, pages 179–196, 2002.

[14] M. Elnozahy, M. Kistler, and R. Rajamony. Energy conservation policies for web servers. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 8–8. USENIX Association, 2003.

[15] K. Govil, E. Chan, and H. Wasserman. Comparing algorithm for dynamic speed-setting of a low-power CPU. In *Proceedings of the Conference on Mobile Computing and Networking*, pages 13–25, 1995.

[16] D. Gross and C. M. Harris. *Fundamentals of queueing theory (2nd ed.).* John Wiley & Sons, Inc., 1985.

[17] R. Guerra, J. Leite, and G. Fohler. Attaining soft real-time constraint and energy-efficiency in web servers. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 2085–2089, 2008.

[18] Y.-T. He. *Exploiting Limited Customer Choice and Server Flexibility.* PhD thesis, McMaster University, 2007.

[19] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini. Energy conservation in heterogeneous server clusters. In *Proceedings of the Symposium on Principles and Practice of Parallel Programming*, pages 186–195, 2005.

[20] A. Iosup, O. Sonmez, S. Anoep, and D. Epema. The performance of bags-of-tasks in large-scale distributed systems. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 97–108, 2008.

[21] O. B. Jennings, A. Mandelbaum, W. A. Massey, and W. Whitt. Server staffing to meet time-varying demand. 42(10):1381–1394, 1996.

[22] A. Kaw and E. Kalu. *Numerical Methods with Applications.* 2008.

[23] D. Kondo, A. A. Chien, and H. Casanova. Resource management for rapid application turnaround on enterprise desktop grids. In *Proceedings of the Conference on Supercomputing*, 2004.

[24] L. Kontothanassis and D. Goddeau. Profile driven scheduling for a heterogeneous server cluster. In *Proceedings of the 34th International Conference on Parallel Processing Workshops*, pages 336–345, 2005.

[25] I. Legrand, H. Newman, R. Voicu, C. Cirstoiu, C. Grigoras, M. Toarta, and C. Dobre. MonALISA: an agent based, dynamic service system to monitor,

control and optimize grid based applications. In *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics*, 2004.

[26] H. Li and R. Buyya. Model-driven simulation of grid scheduling strategies. In *Proceedings of the 3rd International Conference on e-Science and Grid Computing*, pages 287–294, 2007.

[27] C.-H. Lien, Y.-W. Bai, M.-B. Lin, and P.-A. Chen. The saving of energy in web server clusters by utilizing dynamic server management. In *Proceedings of the 12th IEEE International Conference on Networks*, pages 253–257, 2004.

[28] T. Mudge. Power: A first class design constraint. *Computer*, 34:52–57, 2000.

[29] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Dynamic cluster reconfiguration for power and performance. In *Compilers and Operating Systems for Low Power*, pages 75–93. Kluwer Academic Publishers, 2003.

[30] K. Rajamani and C. Lefurgy. On evaluating request-distribution schemes for saving energy in server clusters. In *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 111–122, 2003.

[31] I. Rao and E.-N. Huh. A probabilistic and adaptive scheduling algorithm using system-generated predictions for inter-grid resource sharing. *Journal of Supercomputing*, 45(2):185–204, 2008.

[32] C. Rusu, A. Ferreira, C. Scordino, and A. Watson. Energy-efficient real-time heterogeneous server clusters. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*, pages 418–428, 2006.

[33] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu. Power-aware QoS management in web servers. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, pages 63–72, 2003.

[34] G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, 2009.

[35] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, 1999.