Power-Aware Linear Programming Based Scheduling for Heterogeneous Computer Clusters

Hadil Al-Daoud

Department of Computing and Software
McMaster University

Issam Al-Azzoni

INRIA
Grenoble, France

Douglas G. Down

Department of Computing and Software
McMaster University

Abstract

In the past few years, scheduling for computer clusters has become a hot topic. The main focus has been towards achieving better performance. It is true that this goal has been attained to a certain extent, but on the other hand, it has been at the expense of increased energy consumption and consequent economic and environmental costs. As these clusters are becoming more popular and complex, reducing energy consumption in such systems has become a necessity. Several power-aware scheduling policies have been proposed for homogeneous clusters. In this work, we propose a new policy for heterogeneous clusters. Our simulation experiments show that using our proposed policy results in significant reduction in energy consumption while performing very competitively in heterogeneous clusters.

1. Introduction

Optimizing performance in computer clusters has been a topic of interest in a number of recent research papers. A computer cluster is constructed by networking various machines with different capabilities and coordinating their use to execute a set of tasks. It is true that research has been, to a

certain extent, successful in accomplishing this goal but on the other hand, energy consumption has been mostly neglected.

There is often a trade-off between performance and energy consumption. Thus, good performance can be attained but often 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.* [26]). Also, because of the growing cost of electricity, reducing energy consumption has become an economic necessity (see Bianchini *et al.* [6]). Furthermore, reducing energy consumption helps the environment since gas emissions during electricity production are reduced (see Chase *et al.* [8]).

In this paper, we attempt to 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 [21]). 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 to the state of the system. 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.

In our earlier work ([1] and [2]), we have developed several scheduling policies that 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 paper, 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.

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 paper, 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. These are defined in Section 6.

The organization of the paper is as follows. Section 2 gives the workload model in detail. Section 3 describes several scheduling policies. The Power-Aware LPAS policy is described in Section 4. In Section 5, we present the results obtained in our simulation experiments including simulation results for realistic cluster models. In Section 6, we describe a power-aware scheduling policy for structured systems. Section 7 gives an overview of related work. Section 8 concludes the paper.

A preliminary version of this work appeared in [3].

2. Workload Model

In our model for a computer cluster (see Figure 1), 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 at the 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.* [4]) or parameter-sweep applications (as in Casanova *et al.* [7]). Such

applications are becoming predominant for clusters and grids (see Iosup *et al.* [18] and Li and Buyya [23]).

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 [27] and Seneviratne and Levy [29], for example). The policies considered in this paper 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 [21]). 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 [32] and MonALISA [22] can be used to provide dynamic information on the state of the cluster system. Furthermore, 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. For the time being, we assume that each machine has a single operating frequency for the busy state. We will relax this when we discuss Dynamic Voltage Scaling in Section 5.3. Each machine may have different power consumption 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 $B_j \ll M_{i,j}$. Our power consumption model is

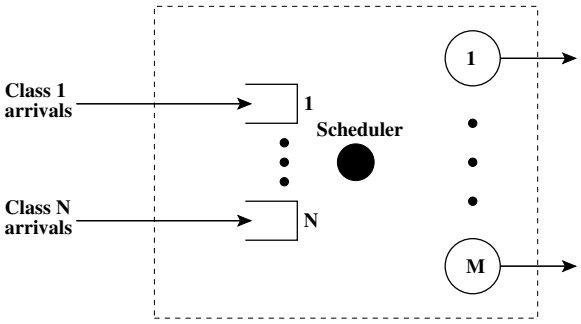

Figure 1: The cluster system model

similar to the one considered in Heath *et al.* [17].

3. Current Policies

A scheduling policy that is applicable to our workload model is the classical First-Come-First-Served (FCFS) policy. FCFS is used in major schedulers (such as Domingues *et al.* [9] and Kondo *et al.* [20]). 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 severe energy wastage.

Another candidate 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. It does not take into account information on the task arrival rates.

4. The 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 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.

$$\begin{aligned}
\max\ & \lambda \\
\text{s.t.}\ & \sum_{j=1}^{J} \theta_{i,j}\mu_{i,j} \geq \lambda\alpha_i, \quad \text{for all } i = 1, \ldots, I, && \text{(1a)} \\
& \sum_{i=1}^{I} \theta_{i,j} \leq 1, \quad \text{for all } j = 1, \ldots, J, && \text{(1b)} \\
& \theta_{i,j} \geq 0, \quad \text{for all } i = 1, \ldots, I, \text{ and } j = 1, \ldots, J. && \text{(1c)}
\end{aligned}$$

The left-hand side of (1a) 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, (1a) 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 (1b) prevents overallocating a machine and (1c) 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 (1). 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. 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 will 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 $i = 1, \ldots, I$, $j = 1, \ldots, J$.

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

$$\text{s.t.} \sum_{j=1}^{J} \delta_{i,j} \mu_{i,j} \geq c\alpha_i, \quad \text{for all } i = 1, \ldots, I, \tag{2a}$$

$$\sum_{i=1}^{I} \delta_{i,j} \leq 1, \quad \text{for all } j = 1, \ldots, J, \tag{2b}$$

$$\delta_{i,j} \geq 0, \quad \text{for all } i = 1, \ldots, I, \text{ and } j = 1, \ldots, J. \tag{2c}$$

The constraint (2a) 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 the energy consumption is minimized while still meeting capacity c.

The Power-Aware 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 (2) always has a solution, since θ^* already satisfies the constraints (2a)-(2c). 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. Note that even though LP (1) may have an infinite number of optimal solutions, the value λ^* is unique and LP (2) only uses λ^*.

In order to achieve the allocations $\delta^*_{i,j}$, we use the following mechanism. Suppose that machine j requests a task at time point 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 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.

The Power-Aware LPAS policy is a dynamic scheduling policy. As the policy only involves solving two LPs, it is suited for scenarios when machines are added and/or deleted from the system or the rates change. On each of these events, one needs to simply solve two new LPs and continue with the new values.

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, the 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 (1) gives $\lambda^* = 1.7647$ and

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

First, set $c = \lambda^*$. Solving LP (2) 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 (2) 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 capacity.

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 1: Simulation Results for Experiment 1

5. Simulation Results

We use simulation to compare the performance of the scheduling policies. In Section 5.1, we simulate artificial systems with different heterogeneities to show the impact of heterogeneity on performance. Then, in Section 5.2, 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, $i = 1, \ldots, I$, $j = 1, \ldots, J$. Note that the distributions are arbitrarily chosen. The Power-Aware policy does not depend on specific distributional assumptions.

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, $M_{i,j}$, $\alpha_{i,j}$, 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 the half width of the interval over the mean value (all statistics are at 95% confidence level).

5.1. 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 [5]). In the first experiment, we simulate a system with high task heterogeneity and high machine heterogeneity. In the second experiment, we simulate a

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 2: Simulation Results for Experiment 2

system with high machine heterogeneity and low task heterogeneity. In both experiments, machine power consumptions are completely heterogeneous.

5.1.1. 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

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

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 (1) gives $\lambda^* = 1.7068$. Table 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 energy saving that is almost 2.5 times that of PME while achieving the same performance. Furthermore, there is still significant energy saving when setting $c = \lambda^*$. This can be explained by the improvement in performance which results in the machines being put into a low power state more often and thereby reducing energy consumption.

5.1.2. 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

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

Note that M in this case was generated randomly, so is different than the M in Section 5.1.1. Our goal is not to compare the two systems.

Solving LP (1) gives $\lambda^* = 1.4582$. Table 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 FIFO 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]).

5.2. Realistic Architectures

In this section, we simulate a system which models a real computer cluster [21] (for details, see He [16]) 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 3. 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. $M_{1,\ldots,10}$ 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.* While the arrival and execution rates are taken from a real system, the machine power consumption matrix, M, is fictitious, as no data is available for these quantities. (It would be useful to have such data to benchmark power saving algorithms.)

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 factor is 6, 4, 7, 5.5, 5, and 6, for group T, U, V, W, X, and Y, respectively.

In both cases, the power consumption in a 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 2 and 3 show the simulation results under both cases (note that the curves for FCFS and PME are virtually identical). The figures show that the Power-Aware LPAS policy performs competitively while reducing

	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 3: The Execution Rates for the System in Section 5.2

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

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

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

Table 4: The Machine Power Consumption Matrix for the System in Section 5.2 - The heterogeneous Case

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 performance.

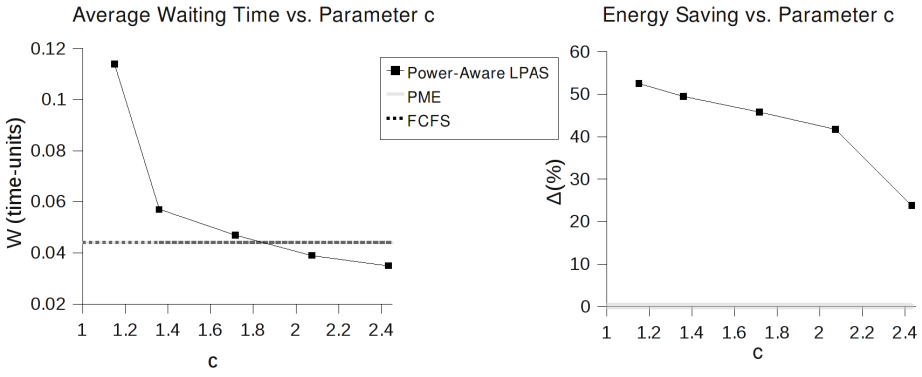

Figure 2: Simulation results for the system in Section 5.2 - The heterogeneous case

5.3. 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 are discussed in Section 7.

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. This policy is used in Govil *et al.* [13] and Rusu *et al.* [28].

In this section, we use the same system specified in Section 5.2 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 [28] 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 5.2. 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 5.2. 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 matrix from Section 5.2. 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 (while this may be overly fast, decreasing the rate would only degrade the performance of Dynamic Voltage Scaling, which makes our comparison conservative). This high frequency allows the machines to quickly adapt their CPU operating frequencies while meeting the load. We performed simulation studies of the two cases in Section 5.2. For the heterogeneous machine power consumption case, using Dynamic Voltage Scaling results in an average task completion 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 completion 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. Thus, the average task completion time increases resulting in an increased energy consumption. This nullifies any energy saving that should be achieved when varying the CPU operating voltages.

6. Structured Systems

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 $\mu_1 = 1$, $\mu_2 = 2$, $\mu_3 = 5$, and $\mu_4 = 3$. Suppose that $\gamma_1 = 1$, $\gamma_2 = 3$, $\gamma_3 = 4$, $\gamma_4 = 0.2$, $\gamma_5 = 6$, $\gamma_6 = 5$, and $\gamma_7 = 10$. Thus, the execution rate matrix is given by:

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

Let $\beta_1 = 3.1$, $\beta_2 = 11.7$, $\beta_3 = 8.2$, $\beta_4 = 6.5$, $\beta_5 = 13.6$, $\beta_6 = 17.4$, and $\beta_7 = 1.3$. Thus, the busy power consumption matrix is given by:

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

Assume also that :

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

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. Table 5 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 ($\frac{1+\lambda^*}{2}$).

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 $\frac{1}{\beta_j}$). 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 (2).

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 (2).*

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 $\sum_{i=1}^{I} \delta^*_{i,j_2} = 0$.

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

$$\sum_{j \neq j_2} \sum_{i=1}^{I} \mu_i \gamma_j \beta_j \delta^*_{i,j}. \tag{3}$$

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 (2a)-(2c).

To show that the constructed solution satisfies (2a), 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 (2b) 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 (2b) holds.

Finally, note that $\overline{\delta^*}_{i,j} \geq 0$ for all j and hence (2c) is satisfied.

The new objective function is given by:

$$\sum_{j\neq j_1,j_2} \sum_{i=1}^{I} \mu_i\gamma_j\beta_j\overline{\delta^*}_{i,j} + \sum_{i=1}^{I} \mu_i\gamma_{j_1}\beta_{j_1}\overline{\delta^*}_{i,j_1} + \sum_{i=1}^{I} \mu_i\gamma_{j_2}\beta_{j_2}\overline{\delta^*}_{i,j_2} \tag{4}$$

By substituting $\overline{\delta^*}_{i,j}, i = 1,\ldots,I, j = 1,\ldots,J$, in (4), 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 $\beta_{j_2}/\beta_{j_1}\leq 1$, it follows that

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

Thus, the corresponding value of the new objective function (4) is no greater than that of (3). Hence, the constructed solution is an optimal solution contradicting our original assumption and the proof is complete. ♢

The following proposition is a direct implication of the lemma.

Proposition 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.*

As a direct implication, we propose a Power-Aware policy that turns on and off machines in the order of β 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) and the threshold (T). The window size determines the decision points. After every WS time units, the scheduler computes the average waiting time for the tasks that are executed during the interval. The parameters W and T determine when a new machine should be added to those being employed or a working machine should be put in

$\frac{1}{\lambda^*}$	1	2	3	4	5	6	7
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	1	0	0.325	1	0	0	1
0.000856	0.75	0	0	0	0	0	1
0.000856	0	0	0	0	0	0	1

Table 5: The Load on each machine for different system loads

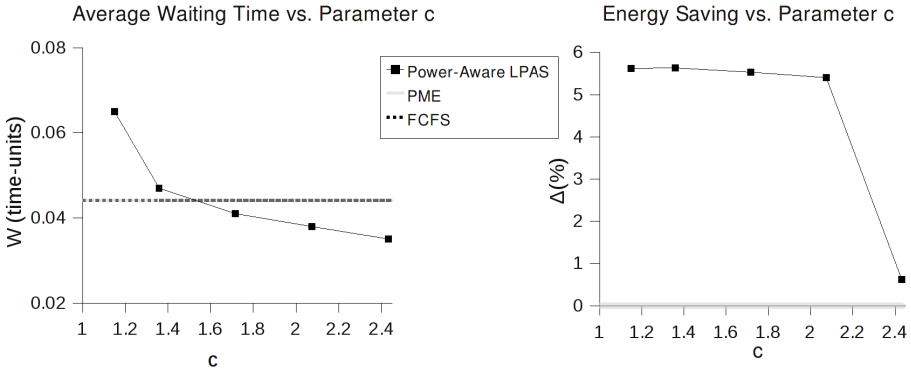

Figure 3: Simulation results for the system in Section 5.2 - The more homogeneous case

a low power state. A new machine is added to those employed when the average waiting time is above $(1-T)W$ and an additional machine is put in a low power state when the average waiting time is below $(1-2T)W$, 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

Policy	c	Δ	W
Ordered-β	-	40.38%	$0.177 \pm 0.33\%$
Power-Aware LPAS	$\lambda^* = 2.3360$	40.93%	$0.167 \pm 0.13\%$
Power-Aware LPAS	midpoint= 1.6680	57.13%	$0.20 \pm 0.32\%$
PME	-	0.009%	$0.164 \pm 0.10\%$
FCFS	-	0%	$0.163 \pm 0.11\%$

Table 6: Simulation Results for the Structured System in Section 6

extremely useful in systems where obtaining such information is difficult or there is a large degree of uncertainty.

Under arrival rates $\alpha = [6.25\ 6\ 6.25\ 6]$, we simulate the structured system defined above using different scheduling policies. The simulation results are given in Table 6. For the ordered-β policy, we use the following values for the parameters: $WS = 25$, $W = 0.2$ and $T = 0.1$. The results show significant energy saving achieved by the ordered-β policy. Furthermore, performance is comparable to that of the Power-Aware LPAS at $c = \lambda^*$ which requires knowledge of the arrival and execution rates as well as the machine power consumptions.

The ordered-β policy also works well for almost structured matrices. For example, consider a variation on the structured system above 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 the interval $[0.5, -0.5]$. One particular pair of execution rate and busy power consumption matrices is given as follows:

$$\mu = \begin{bmatrix} 0.52 & 1.79 & 5.61 & 0.29 & 6.13 & 2.92 & 14.60 \\ 2.74 & 3.04 & 11.04 & 0.24 & 8.42 & 9.73 & 25.91 \\ 7.17 & 15.11 & 21.42 & 0.60 & 34.78 & 34.82 & 57.21 \\ 3.57 & 11.37 & 8.88 & 0.79 & 22.84 & 15.64 & 37.07 \end{bmatrix}$$

and

$$M = \begin{bmatrix} 1.61 & 20.99 & 45.98 & 1.87 & 83.32 & 50.87 & 18.99 \\ 8.49 & 35.52 & 90.55 & 1.55 & 114.57 & 169.24 & 33.68 \\ 22.23 & 176.83 & 175.63 & 3.89 & 473.05 & 605.92 & 74.38 \\ 11.06 & 133.05 & 72.86 & 5.17 & 310.58 & 272.06 & 48.19 \end{bmatrix}$$

respectively.

Policy	c	Δ	W
Ordered-β	-	77.78%	$0.229 \pm 0.78\%$
Power-Aware LPAS	midpoint= 2.0263	74.07%	$0.217 \pm 0.60\%$
FCFS	-	0%	$0.182 \pm 0.14\%$

Table 7: Simulation Results for the Non-Exact Structured System in Section 6

To find the machine power efficiencies (β_j), 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 [31]). 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 [10], 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 service 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 [19]). β_j is set to the slope of the straight line that best fits the data points $(M_{i,j}, \tilde{\mu}_{i,j}), i = 1, \ldots, I$. Several linear regression methods exist. The simplest method is the ordinary least square method which minimizes the sum of squared residuals [19].

By applying singular value decomposition, we get the following:

$$U = \begin{bmatrix} -0.17 & -0.48 & 0.27 & -0.82 \\ -0.30 & -0.81 & -0.19 & 0.47 \\ -0.80 & 0.31 & -0.47 & -0.18 \\ -0.48 & 0.15 & 0.82 & 0.28 \end{bmatrix},$$

$$S = \begin{bmatrix} 99.68 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8.89 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.45 & 0 & 0 & 0 \end{bmatrix},$$

and $V =$

$$\begin{bmatrix} -0.08 & 0.04 & -0.12 & 0.41 & 0 & -0.59 & -0.68 \\ -0.19 & 0.35 & 0.29 & 0.31 & -0.35 & 0.63 & -0.38 \\ -0.26 & -0.40 & -0.50 & -0.50 & -0.20 & 0.25 & -0.42 \\ -0.01 & 0 & 0.06 & -0.01 & 0.90 & 0.32 & -0.28 \\ -0.43 & 0.52 & 0.31 & -0.60 & 0.05 & -0.29 & -0.08 \\ -0.39 & 0.45 & -0.68 & 0.26 & 0.11 & 0.06 & 0.30 \\ -0.74 & -0.49 & 0.30 & 0.26 & 0.06 & -0.05 & 0.21 \end{bmatrix}.$$

Setting

$$\tilde{S} = \begin{bmatrix} 99.68 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

we obtain $\tilde{\mu} = U\tilde{S}V^T =$

$$\begin{bmatrix} 1.39 & 3.12 & 4.26 & 0.16 & 7.04 & 6.44 & 12.27 \\ 2.56 & 5.75 & 7.86 & 0.30 & 12.97 & 11.88 & 22.61 \\ 6.76 & 15.16 & 20.73 & 0.79 & 34.22 & 31.33 & 59.64 \\ 4.07 & 9.13 & 12.49 & 0.47 & 20.61 & 18.87 & 35.92 \end{bmatrix}$$

which is the closest rank 1 matrix to μ. Applying linear regression, we obtain the machine power efficiencies: $\beta_1 = 3.1$, $\beta_2 = 11.7$, $\beta_3 = 8.2$, $\beta_4 = 6.5$, $\beta_5 = 13.6$, $\beta_6 = 17.4$, and $\beta_7 = 1.3$.

The results of simulating this system under arrival rates $\alpha = [6.25\ 6\ 6.25\ 6]$ are shown in Table 7. For the ordered β-policy, the following parameters are used: $WS = 100$, $W = 0.3$ and $T = 0.1$. The results show that the energy saving achieved by the ordered-β policy is still comparable to that achieved by the Power-Aware LPAS policy.

7. Related Work

Energy conservation policies for clusters have been the focus of many researchers. Typically, these policies aim to reduce energy consumption while meeting certain performance requirements. Two basic power management mechanisms dominate the literature: Dynamic Voltage Scaling (DVS) and machine Vary-On/Vary-Off (VOVO).

The Dynamic Voltage Scaling mechanism uses the fact that power consumption P is proportional to the square of the CPU operating voltage V

i.e., $P \propto V^2$ (Elnozahy *et al.* [11] and Mudge [24]). This relation shows that reducing the CPU operating voltage by half will reduce power consumption by a factor of four. 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.,* $V \propto f_{max}$ [11]. By setting the values of the operating voltage and the maximum operating frequency to the lowest values such that the system can still meet performance requirements, the total energy consumption is reduced.

In the machine Vary-On/Vary-Off mechanism, sometimes referred to as the dynamic cluster configuration mechanism, a subset of the available machines are put in a low power state or even completely turned off when the system is not fully utilized [11]. Policies which use the VOVO mechanism should ensure that performance is not severely impacted by keeping enough machines in a normal operation mode.

In this section, we give an overview of several energy conservation policies. First, we look at policies designed for homogeneous clusters. Then, we look at policies designed for heterogeneous clusters.

7.1. Policies for Homogeneous Clusters

Five policies are proposed in [11]. 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 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 threshold we turn a machine on and if it decreases below this threshold then a machine is turned off.

In Pinheiro *et al.* [25], 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.* [30], 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.* [12]. 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 the batching periods. The response time can be controlled by adjusting the batching period.

In Hermenier *et al.* [15], a constraint programming approach is used to determine migration strategies that have as one of their goals to reduce the number of nodes that are required to be busy.

7.2. Policies for Heterogeneous Clusters

The energy conservation policy in [17] 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. Analytical models are required to compute the predicted throughputs and total energy consumption. The Power-Aware LPAS policy does not require such analytical models.

In Rusu *et al.* [28], 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.* [14] 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.

Our work can be seen as taking the VOVO mechanism, but using an LP formulation to initially identify an efficient matching of workload to resources under high system loads. A second LP is then used to identify where VOVO may be used to gain savings as we back off from the maximum system load. This separation of concerns appears to be a novel approach to the problem.

8. Conclusion

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 the machines to minimize the energy consumption. Our simulation results demonstrate that significant energy saving can be achieved if compared to a system that uses 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. In the future, we plan to implement the proposed policies on a real heterogeneous cluster in order to validate the simulation results. We note that there has been little work done on characterizing machine power consumption for heterogeneous systems. Hence, we believe that there is a need to develop a benchmark framework which can be used to compare the different policies in terms of performance and energy consumption. Finally, further investigation on determining (near) optimal values of c would be worthwhile.

References

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

[2] Issam Al-Azzoni and Douglas G. Down. Linear programming-based affinity scheduling of independent tasks on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 19(12):1671–1682, 2008.

[3] Hadil Al-Daoud, Issam Al-Azzoni, and Douglas G. Down. Power-aware linear programming based scheduling for heterogeneous computer clusters. In *Proceedings of The Work in Progress in Green Computing Workshop*, 2010.

[4] 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.

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

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

[7] Henri Casanova, Dmitrii Zagorodnov, Francine Berman, and Arnaud Legrand. Heuristics for scheduling parameter sweep applications in grid environments. In *Proceedings of the 9th Heterogeneous Computing Workshop*, pages 349–363, 2000.

[8] Jeffrey S. Chase and Ronald P. Doyle. Balance of power: Energy management for server clusters. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS)*, pages 163–165, 2001.

[9] Patricio Domingues, Paulo Marques, and Luis 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.

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

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

[12] Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy conservation policies for web servers. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*. USENIX Association, 2003.

[13] Kinshuk Govil, Edwin Chan, and Hal 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.

[14] Raphael Guerra, Julius Leite, and Gerhard Fohler. Attaining soft real-time constraint and energy-efficiency in web servers. In *Proceedings of the Symposium on Applied Computing*, pages 2085–2089, 2008.

[15] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*, pages 41–50, 2009.

[16] Yu-Tong He. *Exploiting Limited Customer Choice and Server Flexibility*. PhD thesis, McMaster University, 2007.

[17] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira Jr., and Ricardo Bianchini. Energy conservation in heterogeneous server clusters. In *Proceedings of the Symposium on Principles and Practice of Parallel Programming*, pages 186–195, 2005.

[18] Alexandru Iosup, Ozan Sonmez, Shanny Anoep, and Dick 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.

[19] Autar Kaw and Egwu Kalu. *Numerical Methods with Applications*. 2008.

[20] Derrick Kondo, Andrew A. Chien, and Henri Casanova. Resource management for rapid application turnaround on enterprise desktop grids. In *Proceedings of the Conference on Supercomputing*, 2004.

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

[22] I.C. Legrand, H.B. 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.

[23] Hui Li and Rajkumar 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.

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

[25] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. Dynamic cluster reconfiguration for power and performance. In *Compilers and Operating Systems for Low Power*, pages 75–93. Kluwer Academic Publishers, 2003.

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

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

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

[29] Sena Seneviratne and David C. Levy. Task profiling for load profile prediction. *Future Generation Computer Systems*, 27(3):245–255, 2011.

[30] Vivek Sharma, Arun Thomas, Tarek Abdelzaher, Kevin Skadron, and Zhijian Lu. Power-aware QoS management in web servers. In *Proceedings of the 24th International Real-Time Systems Symposium*, pages 63–72, 2003.

[31] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2009.

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