Power-Aware Linear Programming Based Scheduling for Heterogeneous Computer Clusters

Hadil Al-Daoud
Department of Computing and Software
McMaster University
Email: aldaouhb@mcmaster.ca

Issam Al-Azzoni
INRIA
Grenoble, France
Email: Issam.Al-Azzoni@imag.fr

Douglas G. Down
Department of Computing and Software
McMaster University
Email: downd@mcmaster.ca

***Abstract*—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.**

***Keywords*-power-aware scheduling; grid computing; linear programming; heterogeneous computing systems;**

I. Introduction

Optimizing performance in computer clusters has been a topic of interest in a number of recent research papers. 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.

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.* [15]). Also, because of the growing cost of electricity, reducing energy consumption has become an economic necessity (see Bianchini *et al.* [5]).

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 VI.

II. Workload Model

In our model for a computer cluster, 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 matrix, 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.* [3]) or parameter-sweep applications (as in Casanova *et al.* [6]).

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 [16]). 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 [12]). 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 [19] and MonALISA [13] 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. 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 $B_{i,j} \ll M_{i,j}$. Our power consumption model is similar to the one considered in Heath *et al.* [10].

III. 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 Kondo *et al.* [11]). 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 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 $M_{i,j}/\mu_{i,j}$. 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.

IV. 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. The goal is to maximize λ such that

$$\sum_{j=1}^{J} \theta_{i,j}\mu_{i,j} \geq \lambda\alpha_i, \qquad \text{for all } i = 1, \ldots, I, \tag{1a}$$

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

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

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 (see [1]) and $\{\theta^*_{i,j}\}$, $i = 1, \ldots, I$, $j = 1, \ldots, J$, 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 $i = 1, \ldots, I$, $j = 1, \ldots, J$.

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

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

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

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

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 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 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.

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 (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.

V. Simulation Results

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, $M_{i,j}$, $\alpha_{i,j}$, 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).

A. 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 [4]). 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.

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

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

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 (1) gives $\lambda^* = 1.7068$. Table I shows the simulation results for the experiment. The table gives

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 I
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 II
SIMULATION RESULTS FOR EXPERIMENT 2

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.

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

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

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

B. Realistic Architectures

In this section, we simulate a system which models a real computer cluster [12] 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 III. 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 IV. $M_{1,\ldots,10}$ 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 $4, \ldots, 9$, *etc.*

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.

	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 III
THE EXECUTION RATES FOR THE SYSTEM IN SECTION V-B

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

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

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

Table IV
THE MACHINE POWER CONSUMPTION MATRIX FOR THE SYSTEM IN SECTION V-B - THE HETEROGENEOUS CASE

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 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 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 1 and 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 performance.

VI. 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 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 :

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

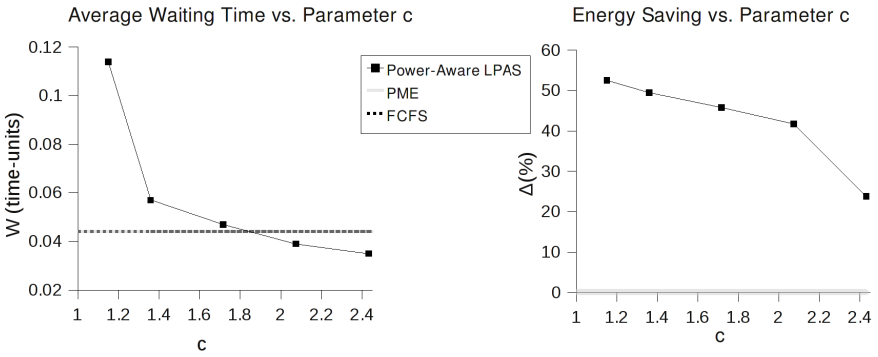

Figure 1. Simulation results for the system in Section V-B - The heterogeneous case

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 V 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 *i.e.*, $\frac{1}{\beta_j}$).

$\frac{1}{\lambda^*}$	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 V
THE LOAD ON EACH MACHINE FOR DIFFERENT SYSTEM LOADS

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 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 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 VI. 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 $\mu_{i,j} = \gamma_j \mu_i (1 + \varepsilon_{i,j}^a)$ and $M_{i,j} = \beta_j \mu_{i,j} (1 + \varepsilon_{i,j}^b)$, $i = 1, \ldots, I$, $j = 1, \ldots, J$. The inaccuracy levels $\varepsilon_{i,j}^a$ and $\varepsilon_{i,j}^b$ are sampled from the uniform distribution $[0.5, -0.5]$. 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

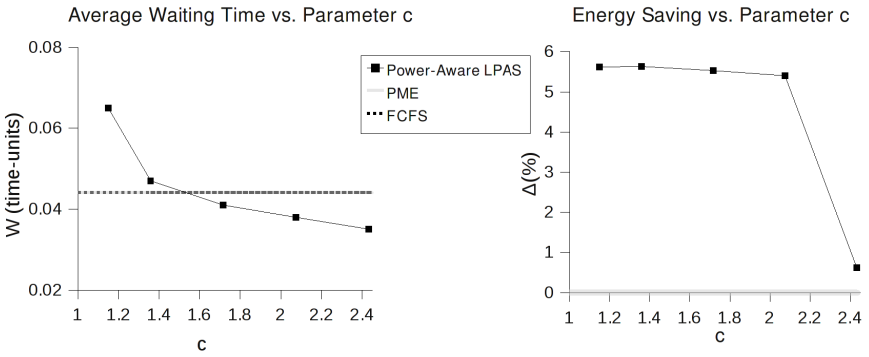

Figure 2. Simulation results for the system in Section V-B - The more homogeneous case

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 VI
SIMULATION RESULTS FOR THE STRUCTURED SYSTEM IN SECTION VI

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

respectively.

The results of simulating this system under arrival rates $\alpha = [6.25\ 6\ 6.25\ 6]$ are shown in Table VII. 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.

VII. 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).

A. Policies for Homogeneous Clusters

Five policies are proposed in [7]. 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. 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.* [14], 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.* [18], 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.* [8]. 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

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 VII
SIMULATION RESULTS FOR THE NON-EXACT STRUCTURED SYSTEM IN SECTION VI

batching periods. The response time can be controlled by adjusting the batching period.

B. Policies for Heterogeneous Clusters

The energy conservation policy in [10] 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.* [17], 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.* [9] 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.

REFERENCES

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

[2] ——, "Linear programming-based affinity scheduling of independent tasks on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 12, pp. 1671–1682, 2008.

[3] 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*, 2006, pp. 56–63.

[4] R. Armstrong, "Investigation of effect of different runtime distributions on SmartNet performance," Master's thesis, Naval Postgraduate School, 1997.

[5] R. Bianchini and R. Rajamony, "Power and energy management for server systems," *Computer*, vol. 37, no. 11, pp. 68–74, 2004.

[6] 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*, 2000, pp. 349–363.

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

[8] 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*. USENIX Association, 2003, pp. 8–8.

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

[10] 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*, 2005, pp. 186–195.

[11] 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.

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

[13] 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.

[14] 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*. Kluwer Academic Publishers, 2003, pp. 75–93.

[15] K. Rajamani and C. 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*, 2003, pp. 111–122.

[16] I. Rao and E.-N. Huh, "A probabilistic and adaptive scheduling algorithm using system-generated predictions for inter-grid resource sharing," *Journal of Supercomputing*, vol. 45, no. 2, pp. 185–204, 2008.

[17] 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*, 2006, pp. 418–428.

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

[19] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *Future Generation Computer Systems*, vol. 15, no. 5-6, pp. 757–768, 1999.