Simulation Modelling Practice and Theory 19 (2011) 239–250

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times

Stylianos Zikos *, Helen D. Karatza

Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

A R T I C L E   I N F O

Article history:
Received 14 May 2010
Received in revised form 16 June 2010
Accepted 17 June 2010
Available online 25 June 2010

Keywords:
Cluster
Local scheduling
Energy aware scheduling
Resource allocation

A B S T R A C T

In this paper we examine three local resource allocation policies, which are based on shortest queue, in a cluster with heterogeneous servers. Two of them are optimized for performance and the third one is optimized for energy conservation. We assume that there are two types of processors in the cluster, with different performance and energy characteristics. We consider that service times of jobs are unknown to the scheduler. A simulation model is used to evaluate the performance and energy behavior of the policies. Simulation results indicate that the differences among the policies depend on system load and there is a trade-off between performance and energy consumption.

1. Introduction

Extensive research has been done on energy conservation and power management techniques for mobile devices to save battery. However, large amounts of energy are consumed at computer centers, clusters and data centers. These entities together can constitute a grid system, such as a cluster grid or a global grid. Therefore, considerable electrical power can be saved with reducing the energy consumption in grid systems. This results not only in the decrease of energy costs but in reduction of green house gas emissions as well (less impact to the environment).

Processors, disks, networks and cooling systems are the main power consumers in these systems. The power consumers just mentioned, have an effect on each other. Their interaction must be taken into account when energy reduction methods are applied, in order to maximize efficiency. For instance, when a processor's clock is reduced, the power of the corresponding cooling system can be also reduced. Another example is when a job replication strategy is applied in order to avoid network activity. In this case hard disk activity would increase. This paper focuses on energy consumed by processors.

In complex distributed systems, such as grids, energy conservation can take place at multiple levels; some examples are server level, cluster level, site level and grid broker level. More particularly, at server level, the energy consumed can be reduced by the use of more energy efficient hardware. Similarly, at cluster level, energy aware job scheduling could save energy. In the present work we focus on cluster-level job scheduling.

Resource heterogeneity is a main characteristic of today's distributed systems, especially of grid systems. Moreover, different resources can co-exist in local single administrative domains, such as clusters. For example, an addition of a bunch of new generation processors could give a boost in overall performance. Another possible situation is the addition of a number of energy efficient processors that belong to the same generation with the preexistent processors, in order to increase efficiency. The issue in these situations is that different types of resources have different power characteristics and power

* Corresponding author. Tel.: +30 6997282768.
E-mail addresses: szikos@csd.auth.gr (S. Zikos), karatza@csd.auth.gr (H.D. Karatza).

1569-190X/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2010.06.009

management features, apart from different performance characteristics. Exploiting heterogeneity to increase energy efficiency is a challenging topic.

Common practices to conserve energy include shutting down idle servers, use of dynamic voltage scaling (DVS) and virtualization. In general, the idle power consumption of a server is about 50–60% of its peak power [1,2]. This fact is a good motive for turning off the idle servers. The advantage is that power consumption is minimized. However, there are some disadvantages. Firstly, a recovery time interval is required in order for the server to be prepared to process jobs (entering active state). Moreover, mechanical hardware equipment, such as hard disks, is stressed due to on–off transitions. Last but not least, turning servers off and on automatically is not a trivial task, especially on heterogeneous platforms. Systems can be entered in sleep mode too. In this case the recovery interval is shortened at the expense of a small amount of energy consumption. In general, the more power you save the longer the latency to return to maximum performance. Supercomputer power states, along with other terms relevant to the Green500 list, are defined in [3]. When multiple servers are running at low utilization, their applications may be consolidated to fewer servers allowing some physical servers to be set to sleep mode. However, the migration costs may overshoot the benefit from the shutdown of servers [4]. In [5], the GREEN-NET framework which is based on an on/off model is presented. Workload prediction and service migration are also taken into account in that work.

Dynamic voltage scaling (DVS) is a power management technique that is used by processors. The voltage is increased or decreased, depending on the system load. When a processor is idle or underutilized, the operating voltage can be reduced in order to save energy. The intuition behind the power savings comes from the basic energy equation that is proportional to the square of the voltage [6]. Therefore, the energy savings are proportional to the square of the voltage reduction. To lower the voltage, the cycle time must be lowered first. While greater reductions can be obtained with slower clocks and lower voltages, computations take longer. Thus, there is a fundamental trade-off between energy and delay [7]. In [8], where a modeling framework is used to estimate the potential of power management schemes to reduce peak power and energy usage, DVS is taken into account as a power management scheme. An energy aware method to schedule multiple real-time tasks in multiprocessor systems that support DVS is presented in [9].

Service virtualization is another approach that increases energy efficiency [10]. Virtualization allows the sharing of hardware. The fact is that many jobs or services often need only a fraction of the available computational resources. Thus, these jobs can be run within a virtual machine (VM) leading to significant increases in overall energy efficiency. Many virtual machines can run on the same hardware system. In this way, the same amount of work can be executed on fewer machines, with increased utilization. Energy consumption at idle machines and consumption due to cooling equipment is reduced, due to the use of fewer machines.

Different approaches have been proposed in order to save energy in data centers and distributed systems. In [11] the authors focus on resource scheduling for parallel scientific workload on multicore-based power-aware clusters, and develop analytical models to approximate performance and energy cost. In [12], the problem of power-aware task scheduling onto heterogeneous and homogeneous multicore processor architectures is addressed. A solution based on game theory is proposed in order to minimize the energy consumption as well as the makespan. The problem of scheduling tasks with dependencies in multiprocessor architectures, where processors have different speeds, is studied in [13]. In that work, an algorithm that post process a schedule of tasks to reduce the energy usage without any deterioration of the makespan, is presented. In [14], the authors propose a framework to simulate energy efficient data grids. Paper [15], provides examples of methods that can greatly reduce electrical power consumption of typical data centers. In [16], energy saving through resource management in a data center is studied. Two resource management techniques are used: dynamic provisioning and load dispatching. Lastly, paper [17], addresses the problem of energy constrained scheduling scheme for the grid environment.

Our previous work [18–21], includes performance evaluation of various grid and local scheduling policies in 2-level computational grid models. A survey about the most important concepts from grid computing related to scheduling problems can be found in [22]. The present paper focuses on resource allocation policies in a cluster. The cluster is heterogeneous regarding the computational capacity of the processors and we assume that scheduler is not aware of job service times. The energy consumption is taken into account in this paper, in addition to performance. On the contrary, in our previous work, heterogeneity was due to the different number of processors, as all processors had exactly the same characteristics. Furthermore, evaluation of the policies was based solely on performance metrics.

The rest of this paper is organized as follows: In Section 2 the system and workload models are described. Section 3 presents the job scheduling policies applied in the cluster. We define the performance and energy metrics, which are used to evaluate the policies, in Section 4. Section 5 presents the simulation setup along with the input parameters. In Section 6 we present and analyze the experimental results. Finally, conclusions and future work are summarized in Section 7.

2. System and workload models

In this paper we model a computational cluster, which could be a part of a computational grid, as an open queueing network. The different local resource allocation policies that we examine are applied to this simulation model, in order to study their performance. The system architecture is depicted in Fig. 1. There is a job arrival stream, with rate λ, towards a local scheduler (LS) whose task is to route an incoming job to a processor. Apart from the LS, the cluster is consisted of a set of 16 servers, each one equipped with its own processor; therefore they can be regarded as independent processing units.

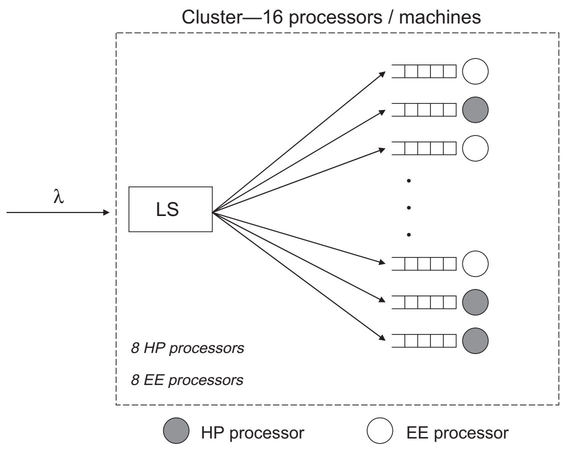

Fig. 1. System architecture.

Table 1
Processor models.

Model	Core (nm)	Voltage (V)	FSB/HT (GHz)	TDP (W)	ACP (W)
Opteron 2386 SE 2.8 GHz	45	1.35	1	137	105
Opteron 2376 HE 2.3 GHz	45	1.2	1	79	55

We must note that the LS and servers are connected via a high speed network as they belong to the same local domain. In each server, jobs can be stored in a local queue whenever the processor is busy. The system is heterogeneous as there are two different classes of processors. Processors from different classes differ in their performance capabilities and power characteristics. More specifically, there are 8 servers with high performance (HP) processors and 8 servers equipped with energy efficient (EE) but slower processors. We outline the characteristics of jobs and processors below.

2.1. Job characteristics

- Any job can be executed on any processor.
- Jobs are nonpreemptable, which means that their execution on a processor can not be suspended until completion.
- Service times of jobs are unknown to LS a priori.
- Jobs are compute-intensive.

2.2. Processor characteristics

- There are two classes of processors which are characterized by
  (1) Different performance capabilities.
  (2) Different power characteristics.

Both the biggest processor manufacturers, AMD and Intel, offer energy efficient chips, apart from their high performance processors. In this study, we have chosen to model the two AMD Opteron[1] processor types that are shown in Table 1. Both processors are quad-core chips and are of the same generation. The Opteron 2386 SE is optimized for performance and represents a HP processor in our model, while the Opteron 2376 HE, which represents an EE processor, is optimized for high efficiency with a lower clock frequency. We have to clarify that both processors include enhanced power management features [23].

Regarding the relative performance between the two processors, we used the following estimation which we made based on benchmarks available at [24,25].

$$\text{Performance ratio} : \frac{\text{HE\_speed}}{\text{SE\_speed}} = 0.8.$$

The benchmarks taken into account are regarding floating point throughput, integer throughput, java server performance and composite theoretical performance (CTP) calculations.

[1] AMD Opteron™ is a trademark of Advanced Micro Devices, Inc.

Table 2
Server power.

Processor type	Active idle (W)	Full load (W)
Opteron 2386 SE 2.8 GHz	125	240
Opteron 2376 HE 2.3 GHz	105	160

Regarding the power characteristics of processors, while processors are serving a job, they operate at full clock speed as we assume that jobs are CPU intensive. Therefore, two processor states are considered in our model: Idle and Full load. When a processor is in idle state (no jobs waiting in queue) the power consumption is reduced due to reduced voltage and frequency (dynamic voltage scaling). This is valid for both processor types. Thus, we have to estimate the power consumption for both processor types in each one of the states. The thermal design power (TDP) is an engineering design specification that is used to represent the maximum power for the processor. However, a processor with a 95 W TDP may not break the 70 W mark under extremely high workloads. The average CPU power (ACP) metric is closer to true power consumption [26] and is used for our estimations in this paper. Our estimations about power consumption were based on Spec[2] results available at [27] and are presented in Table 2. We should note here that the values we chose include all subsystems of the servers and are not referred to processors solely.

Performance per watt is a measure of energy efficiency, as it shows the rate of computation that can be achieved by a computer for every watt of power that it consumes. It can be calculated as the ratio of the performance score from a benchmark to the system power usage. The Green500 list [3] ranks the top 500 supercomputers in the world by energy efficiency using the metric "FLOPS-per-Watt". We next compute the energy efficiency of the two processor types that we model, based on our estimations regarding performance and the power specifications that we presented above.

$$\text{Efficiency\_SE} = \frac{\text{SE\_speed}}{\text{SE\_ACPpower}} = \frac{\text{SE\_speed}}{105} = 0.0095 \times \text{SE\_speed},$$
$$\text{Efficiency\_HE} = \frac{\text{HE\_speed}}{\text{HE\_ACPpower}} = \frac{0.8 \times \text{SE\_speed}}{55} = 0.0145 \times \text{SE\_speed}.$$

From the two above equations we observe that the HE processor, which is represented as EE in our model, shows higher energy efficiency, which is about 50%.

2.3. Distribution of job inter-arrival time and service demand

The inter-arrival times of jobs determine the job arrival rate in the system and therefore, the system load. They are exponential random variables with mean of $1/\lambda$. Job service demand times are also exponential random variables with mean of $1/\mu$. In [28], where a brokering strategy for routing jobs into different sites is presented, the input traffic is a Poisson process and the size of each job is random with an exponential size, in a simple grid queueing model. A stochastic control model for deployment and hosting of a dynamic grid service is introduced in [29], where the arrivals are modelled as a Poisson process due to its practicality, and the executions times are modelled as exponential random variables.

3. Policies

Based on the previously described system model, we conducted simulation experiments in order to evaluate the behavior regarding performance and energy of the three following policies based on shortest queue, which we introduce. We selected the shortest queue policy since it performs well in an environment where job service times are unknown. The policies are applied by the LS in order to allocate jobs to processors and we assume that LS is aware of characteristics of processors in all cases.

3.1. Energy efficiency oriented

3.1.1. Shortest queue with energy efficiency priority (SQEE)

According to the SQEE policy, LS monitors the queue lengths of all processing units, and when a job arrives, it routes the job to the server with the shortest queue. In case there are several empty queues, a server with an idle processor is preferred to a busy server. In case there are:

(a) two idle servers or
(b) different servers of the same queue length,

[2] Spec is trademark of the Standard Performance Evaluation Corporation.

the server with the energy efficient processor is preferred and selected. Thus, SQEE can be characterized as a policy that favors energy efficiency against performance, but only when there is a decision that have to be made (under certain circumstances).

3.2. Performance oriented

3.2.1. Shortest queue with high performance priority (SQHP)

The SQHP is very similar to the policy previously presented, SQEE. The difference is that SQHP selects the server with the high performance processor.

3.2.2. Performance-based probabilistic–shortest queue (PBP–SQ)

The third policy that we introduce, PBP–SQ, is also performance oriented. The main difference as compared to SQHP is that it is based on probabilities. LS groups servers according to their processor characteristics (computational capacity), as the first step. These groups can be viewed or regarded as virtual sub-clusters of the cluster. Therefore, in our model two groups are created, one with the Energy Efficient servers and one consisted of the High Performance servers. In case there are various processor types in a cluster, processors with similar characteristics could be grouped together by the LS. The second step is to define the probabilities for each group. LS utilizes its knowledge about processor performance characteristics and computes the total computational capacity for each group, in order to create the selection probabilities. The probabilities are proportional to the computational capacity. LS exploits the static information about servers to accomplish the two steps mentioned before and is ready to allocate jobs. Upon a job arrival, a group of servers is selected probabilistically and then, the shortest queue policy is applied for the servers that are included in the group. PBP–SQ allocates more jobs to high performance processors due to the higher selection probability.

4. Performance and energy metrics

The parameters, along with performance and energy metrics that are used in simulation computations, are presented in Table 3.

Response time r_i of a job i is the time period from the arrival to the LS to the time service completion of the job. In our model the response time of a job is equal to the delay in a local queue (if any), plus the service time. Let m denotes the total number of processed jobs. The average response time RT is defined as follows [30]:

$$\mathrm{RT} = \frac{1}{m} \times \sum_{i=1}^{m} r_i.$$

Slowdown of a job is the job's response time divided by its service time. If d_i represents the service time of a job i, then the slowdown is defined as follows: $s_i = r_i/d_i$. If m denotes the total number of processed jobs, then the average slowdown SLD equals to:

$$\mathrm{SLD} = \frac{1}{m} \times \sum_{i=1}^{m} s_i.$$

We calculate the energy consumed by the servers, in order to examine the level of energy efficiency that is provided by each one of the three LS policies. The amount of energy used depends on the power and the time for which it is used, according to the following widely-known equation:

$$E = P \times t,$$

Table 3
Notations of the parameters.

P	Number of processors in system
λ	Mean arrival rate
$1/\lambda$	Mean inter-arrival time of jobs
μ	Mean service rate
$1/\mu$	Mean service demand of jobs
U	Average system utilization
RT	Average response time of jobs
DRT	relative increase in RT when SQEE/PBP–SQ is employed instead of SQHP
SLD	Average slowdown
DSLD	Relative increase in SLD when SQEE/PBP–SQ is employed instead of SQHP
TO_E	Total operating energy
TP_E	Total processing energy

where E, P and t, denote energy, power and time, respectively. The standard unit for energy is the joule (J), assuming that power is measured in watts (W) and time in seconds (s). In our simulation model we measure time in time units, therefore the energy is measured in energy units accordingly. Two different energy metrics are taken into account in this paper, the total operating energy (TO_E) and the total processing energy (TP_E), which we define next.

Total operating energy (TO_E) indicates the amount of energy consumed in the cluster from the beginning to the end of simulation. The energy consumption on each one of the 16 servers is computed during simulation and the sum of all servers is calculated at simulation end. Given a time period t, the amount of energy consumed in the time period is depended on two factors: (1) the type of the server (EE or HP) and (2) the server's state (idle or full load). TO_E metric takes into account the energy consumed at idle state, thus, energy consumption increases even if there are no jobs being processed in the system. On the contrary, our second metric total processing energy (TP_E) takes into account only the energy consumption of servers due to job processing. More specifically, it computes the energy consumed by a server only on busy – full load state. When all job processing requests are completed, the sum of all servers is calculated exactly as with the TO_E metric. TP_E is a subset of TO_E.

5. Setup and input parameters

A simulation application, which incorporates the system model, was developed in C programming language to carry out the experiments. The application applies the discrete event simulation technique which is described in [31]. Each simulation experiment ends when 16000 jobs' executions are completed. Experiments showed that longer runs did not affect the results significantly. We used a warm-up period 1000 job executions, in order to make simulation experiments less sensitive to the initial state of the system. Therefore, we start calculating RT and SLD after the 1000th job execution completion. Each result presented is the average value that is derived from 100 simulation experiments with different seeds of random numbers. None noticeable difference in the performance metrics was observed by a further increase in the number of simulation experiments.

5.1. Inter-arrival and service demand times

In this study, we selected the mean service rate (μ_1) of the HP processor to be equal to 1, as a reference. According to the performance ratio among the two processor types presented in Section 2, each EE processor is then characterized by a mean service rate (μ_2) equal to 0.8. Let N_{hp} and N_{ee} be the number of HP and EE processors, respectively. The number of jobs that can be served in one time unit at full load is given by

$$N_{hp} \times \mu_1 + N_{ee} \times \mu_2 = 8 \times 1 + 8 \times 0.8 = 14.4.$$

Therefore, the mean job arrival rate must be less than 14.4 ($\lambda < 14.4$) to maintain the stability of the system.

The system is examined under medium and high load as the following values for mean system utilization were chosen:

$$U = 50\%, 60\%, 70\%, 80\%, 90\%.$$

Using the equation $U = \lambda/\mu$, the job arrival rates derived are the following:

$$\lambda = 7.2, 8.64, 10.08, 11.52, 12.96.$$

The corresponding mean job inter-arrival times that are used as input parameters in the model are:

$$1/\lambda = 0.138, 0.115, 0.099, 0.086, 0.077.$$

We should note here that the average system utilization U values are theoretical, as the actual values depend on the distribution of jobs to processor types.

We selected the mean service demands of jobs to be equal to 1, as an input parameter. However, when referring to the EE processors, the mean job service demands ($1/\mu_2$) increases from 1 to 1.25 due to the reduced processing rate they provide. Table 4 presents the theoretical average system utilization in the case where all 16 processors are of the same type.

6. Experimental results and analysis

In this section we present the simulation results that show the behavior, regarding performance and energy consumption, of the three local resource allocation policies that are examined.

Table 4
Average system utilization.

$1/\lambda$	0.138	0.115	0.099	0.086	0.077
U1	0.45	0.54	0.63	0.72	0.81
U2	0.56	0.68	0.79	0.9	1

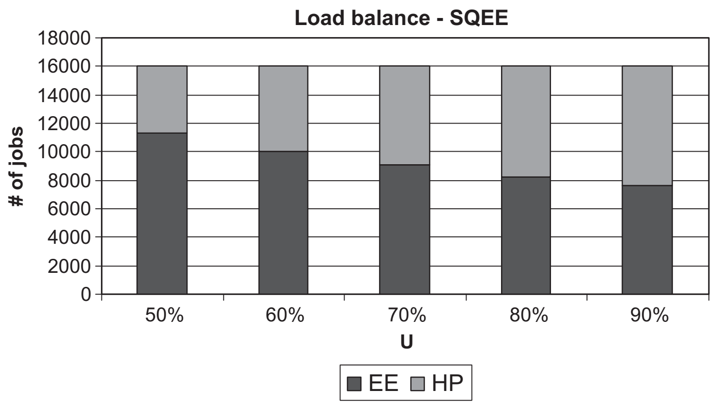

Fig. 2. Number of jobs versus system load when SQEE is employed.

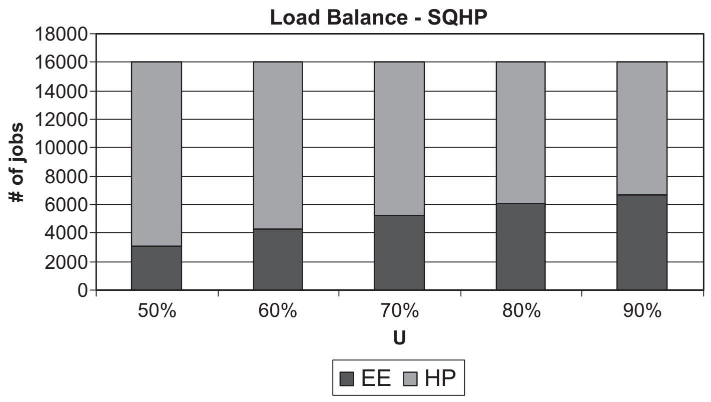

Fig. 3. Number of jobs versus system load when SQHP is employed.

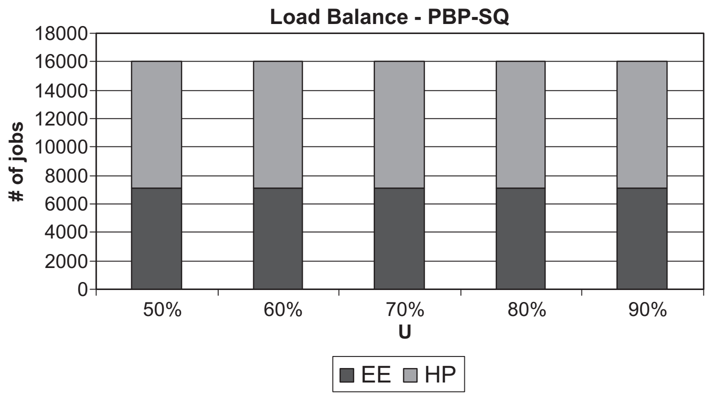

Fig. 4. Number of jobs versus system load when PBP–SQ is employed.

6.1. Load balancing

The average number of jobs distributed to EE and HP servers with regard to system load is shown in Figs. 2–4. In Fig. 2, where the SQEE policy is employed, we observe that for medium utilization more jobs are routed to EE servers. Load balancing among the two processor types is improved with increasing load, as the number of jobs allocated to HP processors increases. This is due to the concentration of jobs to the EE servers.

In the case where the LS operates according to the SQHP policy (Fig. 3), we observe that a large fraction of incoming jobs are routed to HP processors, due to the priority that is given. This is more clearly seen for $U = 50–70\%$. HP processor queues are emptied at a higher rate, on average, due to the higher performance of HP processors. This fact results in SQHP to appear inferior to SQEE regarding to load balancing.

In Fig. 4, where PBP–SQ is utilized, we observe that distribution of jobs to the two different processor types is almost constant for all load values. This is due to the following: When PBP–SQ is applied at the LS, the selection of a processor type is based on probability which depends on the static information of processor performance capability. The selection probabilities are fixed regardless of system load. More jobs are routed to HP processors due to their increased performance. About 7100 jobs are routed to EE processors and 8900 jobs are routed to HP processors. By comparing PBP–SQ with the other two policies, we observe that PBP–SQ balances the load among the processor types more effectively, especially at medium load where $U = 50–60\%$.

6.2. Performance related results

In this subsection we present the performance evaluation in terms of RT and SLD of the three policies, SQEE, SQHP, and PBP–SQ, with regard to system load. RT with regard to system load is depicted in Fig. 5. The first thing to look for in Fig. 5 is that RT increases with increasing load, due to the greater number of jobs in queues and the increased delay in queues. The main point conveyed by Fig. 5 is that the SQHP policy outperforms PBP–SQ and SQEE in all examined load values, because a higher number of jobs are routed to HP processors. For $U = 50\%$, SQEE yields the higher RT due to the vast majority of jobs routed to EE slower processors. For $U > 50\%$, PBP–SQ becomes inferior to SQEE with an increasing difference for U above 80%. This is due to the selection with probabilities method which can lead to a processor assignment where its queue is not the shortest globally.

DRT denotes the relative increase in RT expressed as a percentage, when SQEE and PBP–SQ are utilized instead of SQHP. DRT with regard to system load is presented in Fig. 6. When SQEE is utilized by the LS, DRT decreases with increasing load, ranging between 13% and 5.5%. On the contrary, in the case where PBP–SQ is employed, DRT increases with increasing load, from 10% to 39%. Therefore, the difference in RT between SQEE and SQHP is not very high, as opposed to the difference between PBP–SQ and SQHP which is important, especially at high load.

Fig. 7 presents the average slowdown (SLD) with regard to system load for the three resource allocation policies. First of all, we observe that SLD increases with increasing load, due to the longer delay in local queues. Regarding the relative per-

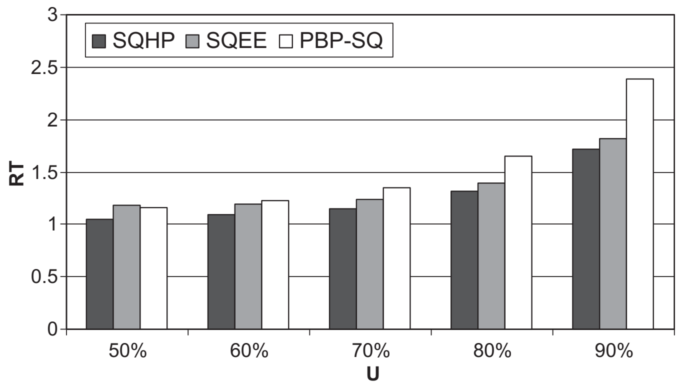

Fig. 5. RT versus system load.

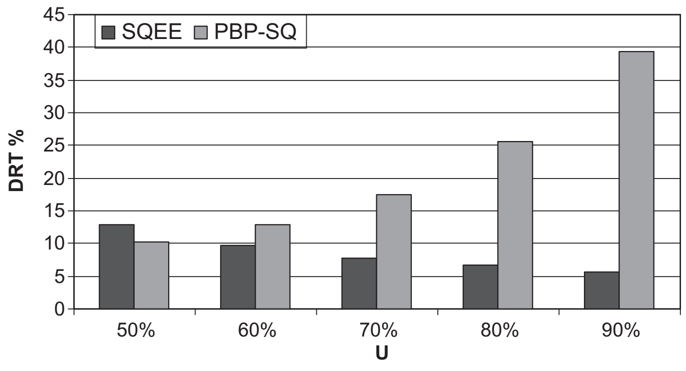

Fig. 6. DRT (%) versus system load.

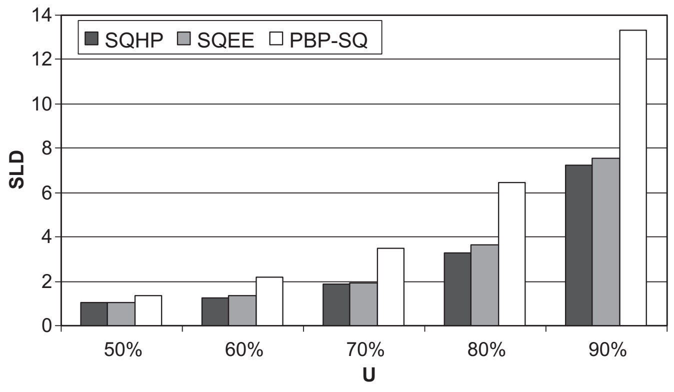

Fig. 7. SLD versus system load.

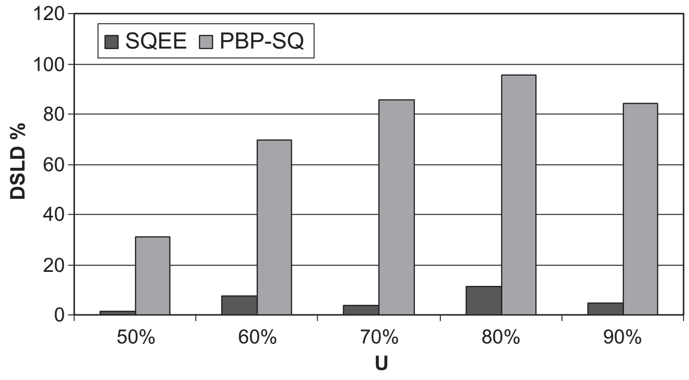

Fig. 8. DSLD (%) versus system load.

formance among the policies, SQHP slightly outperforms SQEE, and PBP–SQ yields the highest SLD. DSLD, which represents the relative increase in SLD when SQEE and PBP–SQ are utilized instead of SQHP, is shown in Fig. 8. The maximum observed DSLD is about 10% when SQEE is utilized, while it is about 95% when PBP–SQ is utilized.

6.3. Energy related results

The total energy consumed by the servers during simulation (TO_E) with regard to system load is presented in Fig. 9. A first observation is that the energy consumed decreases with increasing load, regardless of the applied policy. This is because of the shorter simulation completion time that is observed as load increases. Regarding the scheduling policies, we do not

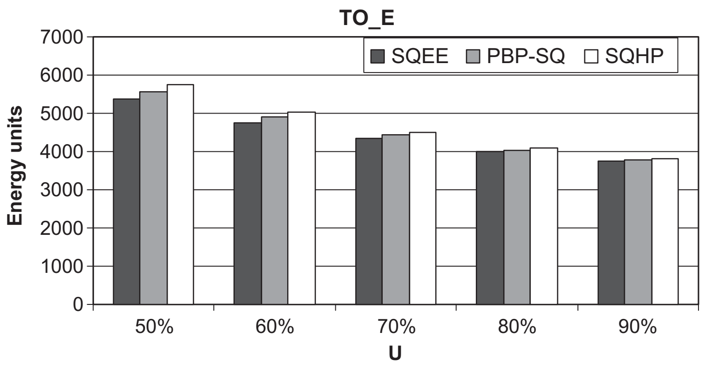

Fig. 9. TO_E versus system load.

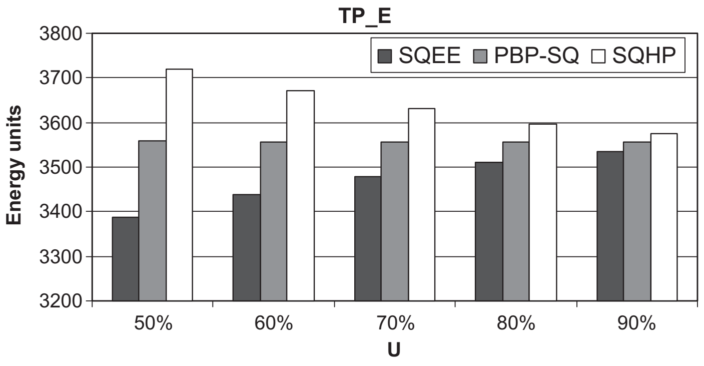

Fig. 10. TP_E versus system load.

notice significant differences between them. Lowest energy consumption is achieved by using the SQEE policy. On the contrary, the highest energy consumption is observed when the SQHP policy is applied. PBP–SQ's curve lies between the two.

Fig. 10 presents the total energy spent for processing jobs (TP_E) with regard to system load. For each load value that is examined, SQHP yields the highest processing energy consumption, followed by PBP–SQ. The lowest processing energy consumption is observed when SQEE policy is applied. Another point is that as load increases, there are three different trends. More specifically, TP_E increases with increasing load when SQEE is employed, while it decreases with increasing load when SQHP is employed. TP_E units consumed are not affected by system load when the PBP–SQ policy is utilized. This behavior is due to the number of jobs routed to EE and HP processors. The result is that difference between SQEE and SQHP decreases as load increases. For example, when $U = 50\%$ there is an increase about 10% in energy consumption when SQHP is employed instead of SQEE. On the other hand, for $U = 90\%$, the increase is limited to 1.13%. By comparing the results of the two energy metrics, TO_E and TP_E (Fig. 9 and 10), we observe that the difference between SQEE and SQHP in the case of TO_E is somewhat smaller.

We thought it would be interesting to investigate the energy consumed by the servers in idle state. Thus, we computed the ratio 'total energy consumed in idle state/total energy consumed', expressed as a percentage. The results are depicted in Fig. 11. We should note here that we examined extra values for system utilization in this case (20%, 30%, and 40%). From the data on the graph, we observe that the contribution of energy consumption in idle state to the total energy consumption decreases with increasing load, regardless of the applied policy. This decrease is expected because as load increases there are less idle servers and simulation time until completion of the workload shortens. By comparing the three resource allocation policies, we practically notice no difference among them at high load (70–90%). When the load is low ($U < 50\%$), we observe that the percentage of energy consumed in idle state is the highest when SQEE is utilized. PBP–SQ shows average consumed energy and SQHP yields the lowest. The difference between the policies is due to the processor selection process. For example, when SQEE is applied at the LS, the utilization of HP processors is lower because EE processors are preferred. Therefore, there are idle HP processors that consume more energy in idle state than EE processors. This could explain the difference that is observed between SQEE and SQHP. Generally, regardless of the policy that is used, the amount of energy consumption in idle state can reach about 70% of the total energy consumption for $U = 20\%$, which is a very high value and is due to underutilization of the system.

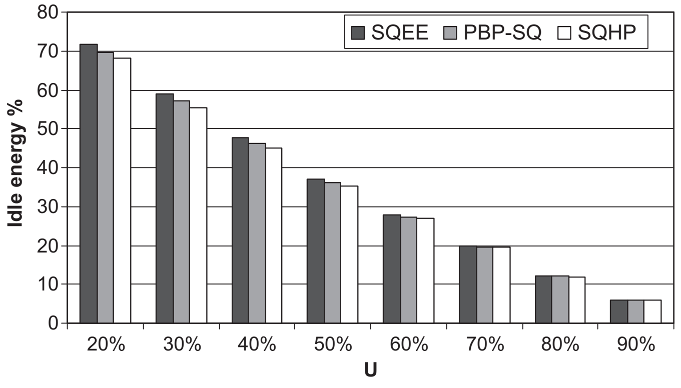

Fig. 11. Total energy consumed at idle state as a percentage versus system load.

Another interesting discussion issue is whether it is beneficial to switch from SQEE to SQHP and vice-versa, depending on the system load. In the case of high load ($U = 90\%$), DRT when SQEE is utilized instead of SQHP is equal to 5.65% (Fig. 6). Furthermore, the relative increase in TO_E when SQHP is utilized instead of SQEE is equal to 1.2%. From the latter fact we observe that at high load the difference between the policies regarding energy consumption is negligible. Therefore, perhaps SQHP would be the preferred policy, in order to avoid the increase in RT by 5.65% which is caused by using SQEE. On the contrary, in the case of medium load ($U = 50\%$), the situation is more complicated. This is due to the increased difference between the policies regarding both performance and energy consumption. The relative increase in RT when SQEE is employed instead of SQHP is larger than the relative increase in TO_E when SQHP is employed instead of SQEE (12.83% versus 6.96%, respectively). However, the selection among SQEE and SQHP when $U = 50\%$ is a priority issue. If we are interested more in energy conservation, SQEE is a good choice; otherwise SQHP should be preferred for its superiority in performance.

7. Conclusions and future work

In this paper we studied three job scheduling policies (SQEE, SQHP, and PBP–SQ) in a cluster. We consider that there are servers with different performance and power characteristics, and the scheduler is not aware of the service times of jobs.

Simulation results showed that each one of the three introduced policies has its own advantages and disadvantages. The SQEE policy, which is energy aware, reduces the energy consumption of the system and yields average job response times. The SQHP policy is optimized for performance, and thus, it outperforms both the other policies, while it yields the highest energy consumption. The differences between SQEE and SQHP decrease with increasing load. Lastly, the PBP–SQ policy performs the worst, especially at high load, and yields average energy consumption. It is the policy which provides the most effective load balance among the two server types, mainly at medium load.

We plan to extend this paper to the case where a computational grid system model is composed of distributed heterogeneous sites with different computational capacity and power characteristics. The objective in this case is to discover and evaluate "green" grid scheduling policies that produce grid job schedules that minimize the energy consumption while they maintain decent performance. An interesting area to extend the research is the cloud computing environment, as well. Another interesting scenario could include the handling of jobs with different orientation. In particular, there may be a number of jobs with fewer performance requirements. Such jobs could be executed on slower resources with lower energy footprint.

References

[1] A. Kansal, F. Zhao, Fine-grained energy profiling for power-aware application design, SIGMETRICS Performance Evaluation Review 36 (2) (2008) 26–31.

[2] L.A. Barroso, U. Holzle, The case for energy-proportional computing, Computer 40 (12) (2007) 33–37.

[3] W. Feng, K. Cameron, Power Measurement of High-End Clusters, The Green500 List, Version 0.1, November 12, 2006.

[4] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of HotPower'08 Workshop on Power Aware Computing and Systems, USENIX, 2008.

[5] G. Da Costa, J. Gelas, Y. Lefevre, A. Orgerie, J. Pierson, O. Richard, K. Sharma, The GREEN-NET framework: energy efficiency in large scale distributed systems, in: Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing (IPDPS), IEEE Computer Society, 2009, pp. 1–8.

[6] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced CPU energy, in: Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, November 14–17, California, 1994.

[7] D. Grunwald, C.B. Morrey, P. Levis, M. Neufeld, K.I. Farkas, Policies for dynamic clock scheduling, in: Proceedings of the 4th Conference on Symposium on Operating System Design and Implementation (OSDI), October 22–25, 2000.

[8] X. Fan, W. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th Annual International Symposium on Computer Architecture, June 9–13, San Diego, CA, USA, 2007, pp. 13–23.

[9] C. Xian, Y. Lu, Z. Li, Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time, in: Proceedings of the 44th Annual Design Automation Conference, June 4–8, San Diego, CA, USA, 2007, pp. 664–669.

[10] A. Berl, E. Gelenbe, M. di Girolamo, G. Giuliani, H. de Meer, M.Q. Dang, K. Pentikousis, Energy-efficient cloud computing, The Computer Journal 19 (2009).

[11] R. Ge, X. Feng, K.W. Cameron, Modeling and evaluating energy-performance efficiency of parallel processing on multicore based power aware systems, in: Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing (IPDPS), IEEE, 2009, pp. 1–8.

[12] I. Ahmad, S. Ranka, S.U. Khan, Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy, in: Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing (IPDPS), IEEE, Miami, 2008, pp. 1–6.

[13] I. Chatzigiannakis, G. Giannoulis, P.G. Spirakis, Energy and Time Efficient Scheduling of Tasks with Dependencies on Asymmetric Multiprocessors. arXiv:0804.4039v2 [cs.DC], 2008.

[14] Z. Zong, X. Qin, X. Ruan, K. Bellam, Y. Yang, A. Manzanares, A simulation framework for energy efficient data grids, in: Proceedings of the 39th Conference on Winter Simulation, IEEE, 2007, pp. 1417–1423.

[15] N. Rasmussen, Implementing Energy Efficient Data Centers, APC White Paper #114, 2006.

[16] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, F. Zhao, Energy-aware server provisioning and load dispatching for connection-intensive internet services, in: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, April 16–18, San Francisco, California, 2008, pp. 337–350.

[17] C. Li, L. Li, Utility-based scheduling for grid computing under constraints of energy budget and deadline, Computer Standards & Interfaces 31 (6) (2009) 1131–1142.

[18] S. Zikos, H.D. Karatza, Resource allocation strategies in a 2-level hierarchical grid system, in: Proceedings of the 41st Annual Simulation Symposium (ANSS), IEEE Computer Society Press, SCS, 2008, pp. 157–164.

[19] S. Zikos, H.D. Karatza, Communication cost effective scheduling policies of nonclairvoyant jobs with load balancing in a grid, The Journal of Systems and Software 82 (12) (2009) 2103–2116.

[20] S. Zikos, H.D. Karatza, The impact of service demand variability on resource allocation strategies in a grid system, ACM Transactions on Modeling and Computer Simulation (TOMACS) 20(4) (2010).

[21] S. Zikos, H.D. Karatza, Clairvoyant site allocation of jobs with highly variable service demands in a computational grid, in: Proceedings of the 9th International Workshop on Performance Modeling, Evaluation, and Optimization of Ubiquitous Computing and Networked Systems (PMEO-UCNS'10), April 19–23, Atlanta, USA (in conjunction with IPDPS 2010), 2010, pp. 1–8.
[22] F. Xhafa, A. Abraham, Computational models and heuristic methods for grid scheduling problems, Future Generation Computer Systems (2010) 608–621.
[23] Quad-Core AMD Opteron Processor Characteristics. <http://www.amd.com/us/products/server/processors/opteron/Pages/opteron-for-server.aspx>, 2010.
[24] AMD Opteron Processor-Based Server Benchmarks. <http://www.amd.com/us/products/server/benchmarks/Pages/benchmarks-filter.aspx>, 2010.
[25] SPEC's Benchmarks and Published Results. <http://www.spec.org/benchmarks.html>, 2010.
[26] AMD White Paper, ACP – The Truth About Power Consumption Starts Here, 2009.
[27] SPEC power_ssj2008 Results. <http://www.spec.org/power_ssj2008/results/>, 2010.
[28] V. Berten, B. Gaujal, Brokering strategies in computational grids using stochastic prediction models, Parallel Computing 33 (4–5) (2007) 238–249.
[29] D. England, J. Weissman, in: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), IEEE, 2004, pp. 192–199.
[30] A. Streit, Enhancements to the decision process of the self-tuning dynP scheduler, in: Job Scheduling Strategies for Parallel Processing Book, Lecture Notes in Computer Science, vol. 3277, Springer, 2005, pp. 63–80.
[31] A. Law, D. Kelton, Simulation Modelling and Analysis, McGraw-Hill, New York, 2006.