A Generalized Model for Investigating Scheduling Schemes in Computational Clusters

Tien V. Do, Binh T. Vu, Xuan T. Tran, Anh P. Nguyen,
T. Krejczinger, N. H. Do, D. Papp, G. Egresits, A. Vitéz

Contents

1 Introduction **3**

2 A Generalized Model and Scheduling Algorithms **6**
2.1 Ranking of Servers 7
2.2 Scheduling 9
2.3 Performance Measures and Energy Metrics 15

3 System Description **18**

4 A Queueing Model and Analysis **20**
4.1 Computation of Eigenvalues and Eigenvectors 25
4.2 Computation of Coefficients 28

5 Results **30**
5.1 Input parameters 31
5.1.1 Parameter of Server Types 31
5.1.2 Load 32
5.2 Numerical results 33
5.2.1 Energy Efficiency 33
5.2.2 Performance 38

6 Conclusion **40**

Abstract

Ebben a beszámolóban egy általánosított modellt mutatunk be számítás igényes feladatok ütemezésének teljesítmény vizsgálatára, ismeretlen kiszolgálási időkkel a számítási klaszterekben. A modellek megalkotásához és a teljesítmény-vizsgálat elvégzéséhez használható paraméterekre a Standard Performance Evaluation Corporation SPECpower ssj2008-as benchmark-jának figyelembe- vételével teszünk javaslatot. Továbbá egy módszert is definiálunk a fizikai szerverek rangsorolására vagy teljesítmény alapú, vagy energiahatékonyság alapú prioritást figyelembe véve, illetve mértékeket, amelyek jellemezhetik a számítási klaszterek teljesítményét. Három sémát vizsgálunk meg (külön sor, osztályozott sor, közös sor) a feladatok pufferelésére a számítási klaszterben, ami szokványos kereskedelmi (Commercial Off-The-Shelf (COTS)) szerverekből épül fel. A numerikus eredmények azt mutatják, hogy a pufferelési sémának nincs hatása a vizsgált klaszter energia felhasználásával kapcsolatos teljesítmény mértékekre, de fontos szerepet játszik az érkező feladatoknál észlelt várakozási és válasz idők alakulásában. Kulcsszavak: heterogén grid modell, pufferelési séma, külön sor, osztályozott sor, közös sor, szerverek rangsorolása

In this paper, we present a generalized model for the performance evaluation of scheduling compute-intensive jobs with unknown service times in computa-

tional clusters. We propose parameters defined and measured according to the `SPECpower_ssj2008` benchmark of the Standard Performance Evaluation Corporation for constructing models and performance investigation. In addition, we also define a method to rank physical servers based on either the high performance priority or the energy efficiency priority, and measures to characterize the performance of computational clusters.

We investigate three schemes (separate queue, class queue and common queue) for buffering jobs in a computational cluster that is built from Commercial Off-The-Shelf (COTS) servers. Numerical results show that the buffering schemes do not have impact on performance measures related to the energy consumption of the investigated cluster. However, the buffering schemes play an important role regarding the waiting time and the response time experienced by arriving jobs.

Keywords: heterogeneous grid model, buffering scheme, separate queue, class queue, common queue, ranking of servers

1 Introduction

The advance of high-speed networking and powerful computers together with the rapid decline of hardware costs led to the widespread application of distributed systems to offer services [1, 2]. In such computational grid systems, job scheduling is multi-criteria

in nature and is a vital task in state-of-the-art resource allocation studies. The rapid increase of grid size and number of users make energy consumption and power capacity a pressing issue, essential to be taken into account in the operation of grid systems.

In the literature, in pursuit of higher performance speed, job allocation algorithms are proposed. These algorithms are applied dominantly at two levels: grid- and cluster-level. Scheduling policies covering and combining both of these levels are listed in [3, 4, 5] and references therein. In addition, some algorithms are implemented with regard to the knowledge about characteristics of jobs. These are classified into clairvoyant [6, 7] and non-clairvoyant algorithms [8, 9].

Nowadays, energy saving has also gained an important role, and optimizing energy consumption has become as crucial as improving performance. The two most commonly used techniques to solve the energy conservation problem are Dynamic Power Management (DPM) [10] and Dynamic Voltage Scaling (DVS) [11]. DPM technique sets the idle components into low-power sleep states and sets back into higher-power active states for operating. But the energy saved in low-power states comes with the drawbacks of energy and time overheads caused by the back and forth transition between sleep and awake states. DVS technique, which is typically supported in the recent CPU of computing elements, uses method of reducing performance speed for power conservation by

lowering the operating voltage/frequency of processing components.

Another approach for the energy consumption problem, which can be used alongside the previous two, is the exploitation of the resource heterogeneity of distributed systems, which include many different types of processors with different power characteristics and performance capacities. This approach has been researched by Zikos and Karatza, who proposed three policies applicable for cluster-level scheduling: SQEE (Shortest Queue based policy with Energy Efficiency priority), SQHP (Shortest Queue based policy with High Performance priority), and PBP-SQ (Performance-Based Probabilistic - Shortest Queue). In their comparison of policies, simulation results indicated that SQEE is the best from the aspect of energy consumption, SQHP outperforms the other two schemes at the price of higher energy consumption, and PBP-SQ proved to be the worst among the three schemes. Note that we have also examined the configuration studied by Zikos and Karatza [12] and can confirm the conclusions of [12]. In addition, Terzopoulos and Karatza [13], Gkoutioudi and Karatza [14] also investigated the scheduling in real-time grid systems.

Compared to previous works [12, 13, 14], this paper provides a generalized model and proposes the parameters defined and measured by Standard Performance Evaluation Corporation. We also define a method to rank physical servers based on either

high performance priority or energy efficiency priority, and measures to characterize the performance of computational clusters. We investigate three schemes (Separate Queue, Class Queue and Common Queue) for buffering jobs in a computational cluster that is built from Commercial Off-The-Shelf (COTS) servers. These proposals allow the systematic (and also other) ways of investigating the performance of computational clusters.

Extensive simulations are performed. Simulation results show that the buffering schemes do not have impact on performance measures related to the energy consumption of the investigated cluster, but are significant factors regarding the waiting time and the response time experienced by arriving jobs.

The rest of the paper is organized as follows. In Section 2, a generalized model along with methods to rank physical servers is proposed. Simulation results are presented in Section 5. Finally, Section 6 concludes the paper.

2 A Generalized Model and Scheduling Algorithms

We consider a computational cluster, which serves compute-intensive jobs. Following [12], we assume that jobs

- can be executed on any server,

- are attended to by the First Come First Served (FCFS) service policy,
- are non-preemptible, which means they cannot be suspended until completion,
- have service times unknown to the local scheduler.

Jobs are to be executed by physical servers according to a specific scheduling algorithm.

In what follows, we describe scheduling algorithms that allocate arriving jobs to a computational cluster. To make the presentation comprehensible, the classification of servers is provided in Section 2.1. Then, the description of scheduling algorithms is given in Section 2.2.

2.1 Ranking of Servers

In a computational cluster, each physical server belongs to the specific type of servers. Let S denote the set of server types and $K = |S|$ be the number of server types. Server type s, $s \in S$, is characterized by two parameters C_s and $P_{ac,s}$, where C_s is the `ssj_ops` value (the number of operations finished during the measurement interval divided by the number of seconds defined for this interval, showing the throughput – workload operations per second– for this period at 100% target load) and $P_{ac,s}$ denotes the average active power measured according to the `SPECpower_ssj2008` benchmark of

the Standard Performance Evaluation Corporation (SPEC) at 100% target load (99.7% actual load).

We introduce two function as follows.

$$r_p(s) = \frac{C_s}{\max\limits_{i \in S} C_i}, \quad s \in S, \tag{1}$$

$$r_e(s) = \frac{\frac{C_s}{P_{ac,s}}}{\max\limits_{i \in S} \frac{C_i}{P_{ac,i}}} \quad s \in S. \tag{2}$$

It is worth emphasizing that $C_s/P_{ac,s}$, $s \in S$ is the performance to power ratio of class s.

Let S_p and S_e denote the ordered sets of server types that are ranked using function (1) and (2), respectively. In ranking S_p (*the high performance priority ranking*) and S_e (*the energy efficiency priority ranking*),

- server types are ranked from 1 to $|S| = K$ based on the computed values $r_p(s)$ and $r_p(s)$, $s \in S$, respectively.
- number one is assigned to server type $\arg \max\limits_s r_p(s)$ and $\arg \max\limits_s r_e(s)$, respectively.

- rank K, assigned to server type arg $\min_s r_p(s)$ and arg $\min_s r_e(s)$, respectively.

For ranking S_p, if there are two server types with the same `ssj_ops` value, the server type of higher average active power gets a higher index.

We organize physical servers based on their type. Physical servers of the same type form a class which is ordered according to either ranking S_p (when *the high performance priority* is chosen) or ranking S_e (when *the energy efficiency priority* is preferred). Physical servers are indexed by pair (i, j), $(i = 1 \ldots, K;\ j = 1, \ldots, M(i))$. Note that server $(1, j)$, $j = 1, \ldots, M(1)$, has the highest priority and server (K, j), $j = 1, \ldots, M(K)$, has the lowest priority.

2.2 Scheduling

The task of scheduling algorithms is to allocate arriving jobs to physical servers. Scheduling algorithms can take into account several factors such as the performance, the power consumption, the number of waiting jobs. Furthermore, the organization of waiting space for jobs that are not immediately served upon their arrival is an important question. In [12], the authors assumed that an arriving job will wait in a specific physical server after the scheduling decision, which is quite straightforward from the aspect of implementation. In this paper, we call this queueing solution as a separate

queue cluster which is illustrated in Figure 1. Furthermore, we also investigate two further schemes for buffering jobs.

- *Separate Queue Cluster*: an arriving job will wait in a specific physical server after the scheduling decision presented in Algorithm 1. As one observes the scheduling algorithm chooses the server with the shortest queue. If there are more idle servers or servers of the same queue length, a server is selected based on the priority of its server type.

- *Class Queue Cluster*: there is a common buffer associated to each class (Figure 2). Jobs scheduled to a specific class wait in the buffer of a specific class when all servers in the specific class are busy. When a job departs from any physical server in the specific class, the first waiting job in the buffer of the specific class will be immediately routed to that server. This procedure is also performed when an arriving job that is routed to the buffer finds an empty server in the specific class. Algorithm 2 routes an arriving job based on criteria: idle servers, the priority and the shortest queue length of classes.

- *Common Queue Cluster*: there is one buffer for storing jobs. On a job arrival, if the LS finds all servers busy, the job will be stored in the common queue and will

wait for an idle server. A job is immediately served if Algorithm 3 finds an idle server upon its arrival.

It is worth emphasizing that the practical implementation of the Class Queue Cluster and the Common Queue Cluster requires the realization of buffers (e.g., waiting jobs stored in the LS), and the signalling procedures to inform the buffers (mechanism that moves a waiting job from buffers to idle servers) about the state of a just freed-up server.

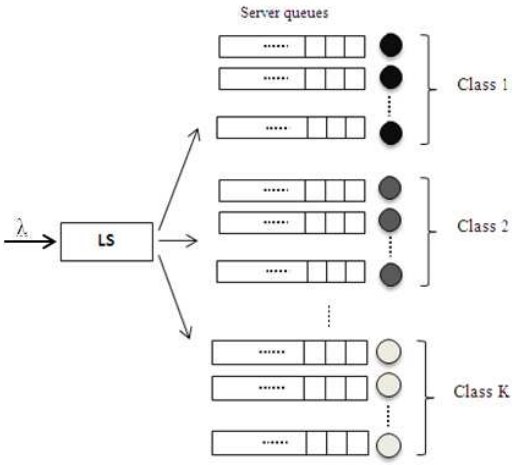


Figure 1: Separate Queue

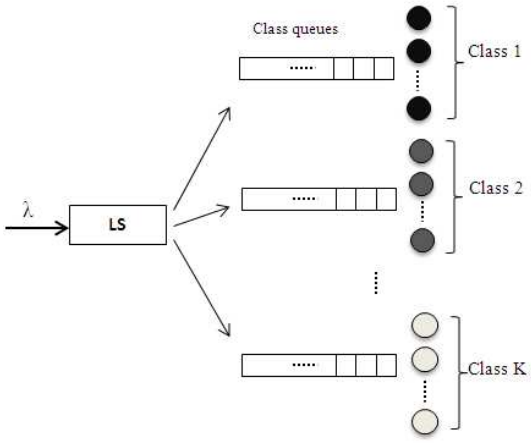


Figure 2: Class Queue

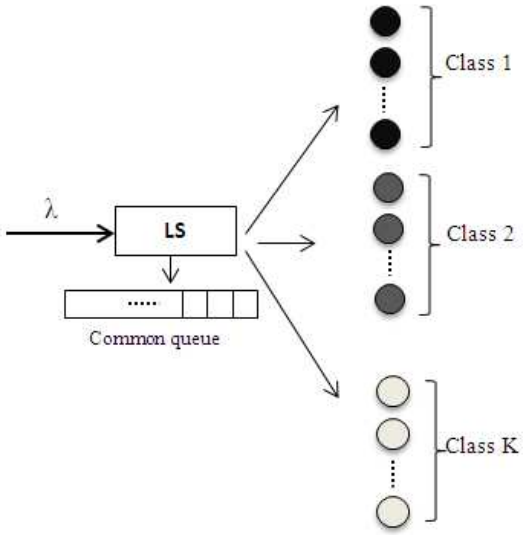


Figure 3: Common Queue

Algorithm 1 The scheduling algorithm for Separate Queue

```
best_server ← (1, 1)
for i = 1 → K do
    for j = 1 → M(i) do
        if server (i, j) is FREE then                    ▷ free server found in class i
            best_server ← (i, j)
            GOTO SCHEDULE
        else
            if queue_length of server (i, j) < queue_length of server best_server
then
                best_server ← (i, j)
            end if
        end if
    end for
end for
SCHEDULE: ROUTE job to queue of server best_server
```

Algorithm 2 The routing algorithm for Class Queue

```
best_class ← 1
for i = 1 → K do
    for j = 1 → M(i) do
        if server (i, j) is FREE then                    ▷ free server found in class i
            best_class ← i
            GOTO SCHEDULE
        else
            if queue_length of class i < queue_length of class best_class then
                best_class ← i
            end if
        end if
    end for
end for
SCHEDULE: ROUTE job to queue of class best_class
```

Algorithm 3 The routing algorithm for Common Queue

```
for i = 1 → K do
    for j = 1 → M(i) do
        if server (i, j) is FREE then                ▷ free server found in class i
            free_server ← (i, j)
            GOTO SCHEDULE
        end if
    end for
end for
SCHEDULE:
if found free_server then
    ROUTE job to queue of class free_server
else
    ROUTE job to Common Queue
end if
```

2.3 Performance Measures and Energy Metrics

Let w_l be the waiting time in queue of job l before service and s_l be the service time that takes the server to process job l. The mean waiting time $WT(n)$ and mean service time of n completed jobs are calculated as follows:

$$WT(n) = \frac{1}{n}\sum_{l=1}^{n} w_l$$

and

$$ST(n) = \frac{1}{n}\sum_{l=1}^{n} s_l.$$

Response time r_l of job l is the time period between the arrival instant and the departure instant of job l. We have $r_l = w_l + s_l$.

The average response time $RT(n)$ of n jobs that finished service is

$$RT(n) = \frac{1}{n}\sum_{l=1}^{n} r_l.$$

The average service time, the average waiting time and the average response time

are defined as

$$\begin{aligned} ST &= \lim_{n\to\infty} ST(n), \\ WT &= \lim_{n\to\infty} WT(n), \\ RT &= \lim_{n\to\infty} RT(n). \end{aligned}$$

Let the departure time of job n be t_n.

Let $\alpha_{i,j}(t)$ denote the total time of active and $\iota_{i,j}(t)$ be the sum of idle time periods of server (i,j) until time t, respectively. If the cluster starts the operation from time instant 0, then $t = \alpha_{i,j}(t) + \iota_{i,j}(t)$.

Since the idle and the active power consumption of server (i,j) are denoted by $P_{id,i}$ and $P_{ac,i}$, the average active energy per job until t_n is

$$AAE(t_n) = \frac{\sum_{i=1}^{K} P_{ac,i} \sum_{j=1}^{M(i)} \alpha_{i,j}(t_n)}{n}, \tag{3}$$

and the average idle energy per job until t_n is

$$AIE(t_n) = \frac{\sum_{i=1}^{N} P_{id,i} \sum_{j=1}^{M(i)} \iota_{i,j}(t_n)}{n}. \tag{4}$$

The long term average active energy per job (AAE) and the long term average idle energy per job (AIE) are defined as

$$AAE = \lim_{n\to\infty} AAE(t_n), \tag{5}$$

$$AIE = \lim_{n\to\infty} AIE(t_n). \tag{6}$$

The long term average energy per job $AOE = AAE + AIE$.

The parameters, along with the performance and energy metrics that are used in simulation computations, are summarized in Table 1.

Table 1: Notations

K	Number of server classes
U	Average system utilization
$WT(n)$	the average waiting time up to job n
WT	the average waiting time
$ST(n)$	the average service time up to job n
ST	the average service time
$RT(n)$	the average response time up to job n
RT	the average response time
$AAE(t)$	the average active energy per job up to time t
AAE	the long term active energy per job
$AIE(t)$	the average idle energy per job up to time t
AIE	the long term idle energy per job

3 System Description

A specific example from the biggest processor manufacturers with specific processors for computational clusters was considered by Zikos and Karatza in [19]. Based on the example, we construct the general architecture for computational clusters as follows.

A computational cluster consists of a number of machines, each is equipped with a processor or processors of a certain performance capability. Let c be the number of machine classes. Note that a machine of class-i, $1 \leq i \leq c$, has performance index $\mathcal{P}_i$ based on benchmarks that are computed based floating point throughput, integer throughput, java server performance and composite theoretical performance (CTP) calculations.

Let M denote the number of machines in the cluster. Let $C(j)$ denote the class of machine j, $1 \leq j \leq M$. Note that $1 \leq C(j) \leq c$.

Compute-intensive jobs arrive are scheduled to machines. 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.

We can have an **intuitive observation** as follows (to be confirmed by the simulation, e.g., [19])

- The number of jobs assigned to and executed in machines is proportional to the

performance index of machines in a long-term.

- The scheduling of jobs follows some cyclic patterns.

The rationale behind this observation is that the utmost goal is to utilize the infrastructure to provide computing service for jobs as well as possible.

Hence, we propose an approximate queueing model for a computational cluster. In the queueing model the allocation of jobs forms a repeated cycle that is designed based on the performance index.

A repeated cycle is partitioned into scheduling lists. Let $L(j,i)$ be the sequence in the ith scheduling list of type j. Let $N(j)$ denote the occurrence number of jobs for server j in a repeated cycle. Let $N(j,i,k)$ represent the occurrence number of jobs for server k in the ith scheduling list of type j for $k \neq j$. Note that $N(j,i,j) = 1$. Furthermore, we introduce $N(j,i,*) = \sum_{l=1}^{h} N(j,i,l)$ and $N = \sum_{i=1}^{N(j)} \sum_{l=1}^{h} N(j,i,l) = \sum_{i=1}^{N(j)} N(j,i,*)$.

We use the term station to refer to a machine or a group of machines of the same category. Assume that we have four stations. Arriving jobs are directed according the following repeated cycle: $\{3 \to 1 \to 3 \to 4 \to 2 \to 3 \to 4 \to 1 \to 2 \to 3 \to 3 \to 4 \to 2\}$. This means, jobs are allocated to station 3, 1, 3, 4,..., 2. Let us consider the second station. Then, there are three scheduling lists for the second station: $L(2,1) = \{3 \to 1 \to 3 \to 4 \to 2\}$, $L(2,2) = \{3 \to 4 \to 1 \to 2\}$ and $L(2,3) = \{3 \to 3 \to 4 \to 2\}$.

Note this is for illustration. We can use the example [19] for the explanation.

4 A Queueing Model and Analysis

Consider an allocation policy that distribute jobs according a repeated cycle. The number of the scheduling lists in the repeated cycle for station j is equal to the occurrence number of jobs assigned to station j in a repeated cycle. Each scheduling list starts from a job allocated to station $j+1$ and ends with a job allocated to station j. Some notations are introduced as follows:

- h is the number of stations ($h = M$ or $h = c$, we will investigate these alternatives).
- the processing times of jobs in station j follows an exponential distribution with rate μ_j.
- jobs arrive according a Poisson process with rate λ.
- $G_{j,i}$ the interarrival time at server j when the sequence follows the ith scheduling list of type j.
- $G_{j,i}(t)$ distribution function of $G_{j,i}$, i.e., $G_{j,i}(t) = \Pr(G_{j,i} < t)$.
- $G^*_{j,i}(s)$ Laplace-Stieltjes transform $G^*_{j,i}(s) = \int_0^\infty e^{-st} dG_{k,i}(t)$.

- $p_{k,i,j}$ the probability that the number of jobs completed in station j during $G_{k,i}$ is equal to k when the production sequence follows the ith scheduling list of type j.

- $\pi_{n,i,j}$ the stationary probability that the number of jobs remaining in station j is equal to n at the moment a job arrives at station j and the production sequence follows the ith schedule list.

Let $\aleph_m$ denote the number of jobs in station j and $\beth_m$ be the scheduling list seen by the mth arrival of a job at station j.

Note that $\{\aleph_m, I_m, m \geq 1\}$ is an embedded Markov chain on state space $\{(n,i)|n = 0,1,\ldots,\infty; i = 1,2,\ldots,N(j)\}$. Let $\pi_{n,i,j}$ denote the stationary probability as $\pi_{n,i,j} = \Pr_{m\to\infty}(\aleph_m = n, I_m = i)$. We introduce vector $\boldsymbol{\alpha}_{n,j} = (\pi_{n,1,j},\ldots,\pi_{n,N(j),j})$.

We can write the following equations

$$G^*_{j,i}(s) = \frac{\lambda}{\lambda+s}\prod_{k=1,\neq j}^{h}\left(\frac{\lambda}{\lambda+s}\right)^{N(j,i,k)} = \left(\frac{\lambda}{\lambda+s}\right)^{\sum_{k=1}^{h} N(j,i,k)} = \left(\frac{\lambda}{\lambda+s}\right)^{N(j,i,*)}, \quad (7)$$

$$p_{k,i,j} = \int_0^\infty \frac{(\mu_j t)^k e^{-\mu_j t}}{k!} dG_{j,i}(t) = \frac{(-\mu_j)^k}{k!} G^{*(k)}_{j,i}(\mu_j), \quad (8)$$

where $G^{*(k)}_{j,i}(s)$ is the order-k derivative of $G^*_{j,i}(s)$ with respect to s.

For $n \geq 0$, matrix $\mathbf{A}_n$ is introduced as follows:

$$\mathbf{A}_n = \begin{bmatrix} 0 & p_{n,2,j} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & p_{n,3,j} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & & \dots & 0 & 0 & p_{n,N(j),j} \\ p_{n,1,j} & 0 & & \dots & 0 & 0 & 0 \end{bmatrix}, \ (n \geq 0).$$

Let us define $b_{n,i,j} = 1 - \sum_{l=0}^{n-1} p_{l,i,j}$ for $n = 1, 2, \dots$ and introduce vectors

$$\mathbf{B}_n = \begin{bmatrix} 0 & b_{n+1,2,j} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & b_{n+1,3,j} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & & \dots & 0 & 0 & b_{n+1,N(j),j} \\ b_{n+1,1,j} & 0 & & \dots & 0 & 0 & 0 \end{bmatrix}, \ (n \geq 0).$$

The transition matrix of the embedded Markov chain can be written as follows

$$\begin{bmatrix} \mathbf{B}_0 & \mathbf{A}_0 & 0 & 0 & 0 & 0 & \dots \\ \mathbf{B}_1 & \mathbf{A}_1 & \mathbf{A}_0 & 0 & 0 & 0 & \dots \\ \mathbf{B}_2 & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & 0 & 0 & \dots \\ \mathbf{B}_3 & \mathbf{A}_3 & \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}. \tag{9}$$

The balance equations are written as follows:

$$\boldsymbol{\alpha}_{0,j} = \sum_{n=0}^{\infty} \boldsymbol{\alpha}_{n,j} \mathbf{B}_n, \tag{10}$$

$$\boldsymbol{\alpha}_{n,j} = \sum_{k=-1}^{\infty} \boldsymbol{\alpha}_{n+k,j} \mathbf{A}_{k+1} \ \ (n \geq 1). \tag{11}$$

In addition, the normalization equation is

$$\sum_{i=1}^{N(j)} \sum_{n=0}^{\infty} \pi_{n,i,j} = 1. \tag{12}$$

Let $\mathbf{Q}(x) = \mathbf{A}_0 + (\mathbf{A}_1 - \mathbf{I})x + \sum_{k=2}^{\infty} \mathbf{A}_k x^k$ be the characteristic matrix polynomial. Assume that $\mathbf{Q}(x)$ has d pairs of eigenvalue-eigenvectors $(x_i, \boldsymbol{\psi}_i)$, thus satisfying the equations

$$\boldsymbol{\psi}_i \mathbf{Q}(x_i) = 0, \quad \text{for } i = 1, \ldots, d, \tag{13}$$

$$det[\mathbf{Q}(x_i)] = 0, \quad \text{for } i = 1, \ldots, d. \tag{14}$$

Following [18], for the kth $(k = 1, \ldots, d)$ eigenvalue-eigenvector pair, $(x_k, \boldsymbol{\psi}_k)$, by substituting $\boldsymbol{\alpha}_{n,j} = \boldsymbol{\psi}_k x_k^n$ in the equations (11), this set of equations is satisfied. This means $\boldsymbol{\alpha}_{n,j} = \boldsymbol{\psi}_k x_k^n$ is a particular solution. As a consequence, the general solution for $\boldsymbol{\alpha}_{n,j}$ is a linear sum of all the factors $(\boldsymbol{\psi}_k x_k^n)$. That is, we can write

$$\boldsymbol{\alpha}_{n,j} = \sum_{k=1}^{d} \beta_k x_k^n \boldsymbol{\psi}_k \; (n \geq 0), \tag{15}$$

where $(x_1, \boldsymbol{\psi}_1)$, ..., $(x_d, \boldsymbol{\psi}_d)$ are left-hand-side eigenvalue-eigenvector pairs of the characteristic matrix polynomial $\boldsymbol{Q}(x)$ and $\beta_1, \ldots, \beta_d$ are coefficients to be determined. In order to satisfy the normalization equation (12), only $|x_i| < 1$ should be considered. Without the loss of generality, let d be the number of these eigenvalues x_i. The number of coefficients $\beta_1, \ldots, \beta_d$ to be determined is d. We can utilize equation (10) and

the normalization equation to compute the coefficients. Thus, the number of linear equations is $N(j)+1$. However, only $N(j)$ linear equations are linearly independent. Therefore, the linear equations have the unique solution if and only if $N(j)=d$. This means, the exact number of eigenvalues inside the unit circle (i.e., $|x_i|<1$) is $N(j)$.

Therefore, we have to determine the eigenvalues, the eigenvectors and the coefficients for the stationary probabilities.

4.1 Computation of Eigenvalues and Eigenvectors

We have

$$\begin{aligned}\sum_{k=0}^{\infty} x^k p_{k,i,j} &= \sum_{k=0}^{\infty} \int_0^{\infty} \frac{(x\mu_j t)^k e^{-\mu_j t}}{k!} dG_{j,i}(t) \\ &= \int_0^{\infty} \left(\sum_{k=0}^{\infty} \frac{(x\mu_j t)^k}{k!} \right) e^{-\mu_j t} dG_{j,i}(t) \\ &= \int_0^{\infty} e^{x\mu_j t} e^{-\mu_j t} dG_{j,i}(t) \\ &= G^*_{j,i}(\mu_j(1-x)). \end{aligned} \tag{16}$$

Based on equation (16) we obtain the closed form for the characteristic matrix

polynomial as

$$\mathbf{Q}(x) = \mathbf{A}_0 + (\mathbf{A}_1 - \mathbf{I})x + \sum_{k=2}^{\infty} \mathbf{A}_k x^k = -\mathbf{I}x + \sum_{n=0}^{\infty} \mathbf{A}_n x^n,$$

$$\mathbf{Q}(x) = \begin{bmatrix} -x & \sum_{n=0}^{\infty} p_{n,2,j} x^n & 0 & \dots & 0 & 0 & 0 \\ 0 & -x & \sum_{n=0}^{\infty} p_{n,3,j} x^n & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & & \dots & 0 & -x & \sum_{n=0}^{\infty} p_{n,N(j),j} x^n \\ \sum_{n=0}^{\infty} p_{n,1,j} x^n & 0 & & \dots & 0 & 0 & -x \end{bmatrix} =$$

$$\begin{bmatrix} -x & G^*_{j,2}(\mu_j(1-x)) & 0 & \dots & 0 & 0 & 0 \\ 0 & -x & G^*_{j,3}(\mu_j(1-x)) & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & & \dots & 0 & -x & G^*_{j,N(j)}(\mu_j(1-x)) \\ G^*_{j,1}(\mu_j(1-x)) & 0 & & \dots & 0 & 0 & -x \end{bmatrix}.$$

The eigenvalues x_i's $(i = 1, \dots, N(j))$ inside the unit circle can be easily obtained because of the closed form of the characteristic matrix polynomial $\mathbf{Q}$.

Because of the special structure of the characteristic matrix polynomial $\mathbf{Q}$ we obtain

$$det[\mathbf{Q}(x)] = (-x)^{N(j)} - (-1)^{N(j)} \prod_{i=1}^{N(j)} G^*_{j,i}(\mu_j(1-x)). \tag{17}$$

Substituting (7) into equation (17), we get

$$\begin{aligned} det[\mathbf{Q}(x)] &= (-x)^{N(j)} - (-1)^{N(j)} \prod_{i=1}^{N(j)} \left(\frac{\lambda}{\lambda + \mu_j(1-x)}\right)^{N(j,i,*)} \\ &= (-x)^{N(j)} - (-1)^{N(j)} \left(\frac{\lambda}{\lambda + \mu_j(1-x)}\right)^{\sum_{i=1}^{N(j)} N(j,i,*)} \\ &= (-x)^{N(j)} - (-1)^{N(j)} \left(\frac{\lambda}{\lambda + \mu_j(1-x)}\right)^{N}. \end{aligned} \tag{18}$$

Since there are various technique to find the roots of polynomials, we can easily find the roots of $det[\mathbf{Q}(x)] = 0$ and therefore the eigenvalues (x_i's for $i = 1, \ldots, N(j)$ inside the unit circle) of the characteristic matrix polynomial $\mathbf{Q}$ as well. Then, eigenvectors are calculated as follows. Let $\boldsymbol{\psi}_i = [\psi_{i,1}, \psi_{i,2}, \ldots, \psi_{i,N(j)}]$ be an eigenvector for eigenvalue x_i ($i = 1, 2, \ldots, N(j)$). Expanding equation (13), we get the recursive relations $\psi_{i,l+1} = \psi_{i,l} G^*_{j,l+1}/x$ between $\psi_{i,l}$ and $\psi_{i,l+1}$ for $l = 1, \ldots, N(j) - 1$. An eigenvector remains as the eigenvector corresponding to the same eigenvalue when it is multipled by a scalar. Using this property, we can determine $\boldsymbol{\psi}_i$ by setting $\psi_{i,1} = 1$ and using the above recursive relations to compute $\psi_{i,l}$ for $l = 2, \ldots, N(j)$.

4.2 Computation of Coefficients

Let $\mathbf{E}$ be a matrix

$$\mathbf{E} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & & \dots & 0 & 0 & 1 \\ 1 & 0 & & \dots & 0 & 0 & 0 \end{bmatrix}.$$

Since $b_{n,i,j} = 1 - \sum_{l=0}^{n-1} p_{l,i,j}$ for $n = 1, 2, \dots$ by definition, $\mathbf{B}_n + \sum_{i=0}^{n} \mathbf{A}_i = \mathbf{E}$ holds.

Therefore, we obtain

$$\begin{aligned} &\sum_{n=0}^{\infty} \boldsymbol{\alpha}_{n,j} \left(\mathbf{B}_n + \sum_{i=0}^{n} \mathbf{A}_i \right) = \sum_{n=0}^{\infty} \boldsymbol{\alpha}_{n,j} \mathbf{E} \\ &= \sum_{n=0}^{\infty} \sum_{k=1}^{d} \beta_k x_k^n \boldsymbol{\psi}_k \mathbf{E} = \sum_{k=1}^{d} \frac{\beta_k}{1 - x_k} \boldsymbol{\psi}_k \mathbf{E}. \end{aligned} \tag{19}$$

We can derive the following relation

$$\begin{aligned} &\sum_{n=0}^{\infty} \boldsymbol{\alpha}_{n,j} \sum_{i=0}^{n} \mathbf{A}_i = \sum_{i=0}^{\infty} \sum_{n=i}^{\infty} \boldsymbol{\alpha}_{n,j} \mathbf{A}_i = \sum_{k=1}^{d} \sum_{i=0}^{\infty} \sum_{n=i}^{\infty} \beta_k x_k^n \boldsymbol{\psi}_k \mathbf{A}_i \\ &= \sum_{k=1}^{d} \frac{\beta_k}{1 - x_k} \boldsymbol{\psi}_k \sum_{i=0}^{\infty} \mathbf{A}_i x_k^i = \sum_{k=1}^{d} \frac{\beta_k}{1 - x_k} \boldsymbol{\psi}_k (\mathbf{Q}(x_k) + x_k \mathbf{I}), \end{aligned} \tag{20}$$

where $\mathbf{I}$ is the identity matrix of size $N(j)$. Based on equations (19) and (20) we obtain

$$\sum_{n=0}^{\infty} \boldsymbol{\alpha}_{n,j} \mathbf{B}_n = \sum_{k=1}^{d} \frac{\beta_k}{1-x_k} \boldsymbol{\psi}_k \mathbf{E} - \sum_{k=1}^{d} \frac{\beta_k}{1-x_k} \boldsymbol{\psi}_k (\mathbf{Q}(x_k) + x_k \mathbf{I}). \tag{21}$$

Using (21), equation (10) can be rewritten as follows

$$\boldsymbol{\alpha}_{0,j} = \sum_{k=1}^{d} \frac{\beta_k}{1-x_k} \boldsymbol{\psi}_k \mathbf{E} - \sum_{k=1}^{d} \frac{\beta_k}{1-x_k} \boldsymbol{\psi}_k (\mathbf{Q}(x_k) + x_k \mathbf{I}). \tag{22}$$

The normalization equation can be rewritten as follows

$$\sum_{n=0}^{\infty} \sum_{k=1}^{d} \beta_k x_k^n \boldsymbol{\psi}_k \boldsymbol{e} = \sum_{k=1}^{d} \frac{\beta_k}{1-x_k} \boldsymbol{\psi}_k \boldsymbol{e} = 1, \tag{23}$$

where $\boldsymbol{e}$ is a vector of size $N(j)$ with all elements equal to 1. To compute the coefficients $\beta_1, \ldots, \beta_{N(j)}$, we have to utilize equations (22) and (23), which form the system of $N(j)$ linear equations out of $N(j)+1$ equations (because only $N(j)$ equations are independent).

Several performance measures can be obtained as

- The mean response time of station-j can be expressed as

$$D_j = \frac{\sum_{n=0}^{\infty}(n+1)(\boldsymbol{\alpha}_{n,j}\mathbf{e_j})}{\mu_j} = \frac{\sum_{n=0}^{\infty}(n+1)(\sum_{k=1}^{N(j)}\beta_{k,j}x_{k,j}^{n}\boldsymbol{\psi}_{k,j}\mathbf{e_j})}{\mu_j}$$
$$= \frac{\sum_{k=1}^{N(j)}\beta_{k,j}\sum_{n=0}^{\infty}(n+1)x_{k,j}^{n}\boldsymbol{\psi}_{k,j}\mathbf{e_j}}{\mu_j} = \frac{\sum_{k=1}^{N(j)}\frac{\beta_{k,j}}{(1-x_{k,j})^2}\boldsymbol{\psi}_{k,j}\mathbf{e_j}}{\mu_j}, \quad (24)$$

  where $\boldsymbol{e}_j$ is a vector of size $N(j)$ with all elements equal to 1.

- The average number of waiting jobs

5 Results

Due to the flexibility concerning the modification of codes, we have chosen the SimPack toolkit [15] for all our simulation studies. We have implemented the system models and scheduling algorithms in Section 2. In what follows, if it is not stated otherwise, our simulations are performed with the confidence level of 99%. The accuracy (i.e. the ratio of the half-width of the confidence interval and the mean of collected observations) of *AAE*, *AIE*, *AOE* is of 10^{-5}, while the accuracy of waiting times is less than 0.001.

5.1 Input parameters

5.1.1 Parameter of Server Types

We assume that the considered computational cluster is built from three types of Commercial Off-The-Shelf (COTS) servers that are listed in Table 2.

Table 2: Server specifications

Server type	C_s	$P_{ac,*}$ (W)	$C_s/P_{ac,*}$	$P_{id,*}$ (W)
PRIMERGY RX300 S4 (INTEL Xeon E5440) [?]	303846	265	1147	166
PRIMERGY TX300 S4 (Intel Xeon L5430) [?]	293162	220	1333	138
PRIMERGY TX150 S6 (Intel Xeon X3360) [?]	184145	116	1587	59.3

The computing capability (performance) and the power consumption of three server types based on the specification `SPECpower_ssj2008` of SPEC [16] are also presented in Table 2. Note that the computing capability ssj_ops [17] (number of operations finished per second) and the power consumption $P_{ac,*}$ are measured at 100% target load, while $P_{id,*}$ is the power consumption when the servers are in the idle state. The computed rankings based on the rule in Section 2.1 are reported in Table 3.

Table 3: Ranking function

Server type	**Ranking based on Performance**	**Ranking based on Energy Efficiency**
RX300	$r_p(1) = 1.0$	$r_e(1) = 1147/1587 \approx 0.723$
TX300	$r_p(2) = 293162/303846 \approx 0.96$	$r_e(2) = 1333/1587 \approx 0.840$
TX150	$r_p(3) = 184145/303846 \approx 0.60$	$r_e(3) = 1.0$

5.1.2 Load

Job arrivals follow an exponential distribution with λ parameter. The execution times of jobs are also exponentially distributed. Each job requires a computing capacity equivalent to 303846 (ssj_ops) in average, which means the average execution time of jobs is one second if jobs are routed to a server of type RX300. Let μ_1, μ_2 and μ_3 denote the service rates, if jobs are scheduled to a server of type RX300, TX300 and TX150, respectively. Then we obtain $\mu_1 = 1/s$, $\mu_2 = 0.96/s$ and $\mu_3 = 0.6/s$ as a consequence of the assumption of the average computing capacity required by jobs and Table 2.

We investigate a computational cluster with three types of COTS servers and eight physical machines in each server class. Therefore, the total `ssj_ops` of the computational cluster is $8 * (303846 + 293162 + 184145) = 6249224$.

The investigation is carried out with load U equal to 50%, 60%, 70%, 80% and 90%, where $U = \lambda/(8 * (303846 + 293162 + 184145))/303846 = \lambda/(8 * (1 + 0.96 + 0.6))$. Therefore, the mean inter-arrival time of jobs in our simulation study takes the values

of 0.098 s , 0.081 s, 0.070 s, 0.061 s and 0.054 s, respectively.

5.2 Numerical results

In this section, we present our experimental results of the three system models examined when applying two scheduling policies based on Energy Efficiency (EE) priority and High Performance (HP) priority, regarding their energy consumption and performance capacity behavior.

5.2.1 Energy Efficiency

This subsection presents the energy consumption of the models illustrated by the energy parameters introduced in the previous section: AAE (Average Active Energy), AIE (Average Idle Energy) and AOE (Average Operating Energy). Figures 4 - **??** depict the energy measurements taken at varying system load for the three scenarios under the applied policies.

It can be observed that the average idle energy always shows a decreasing tendency, since a higher system load brings about a higher job demand, which require more active service time and leaves less idle time. The less time the servers spend in the idle state,

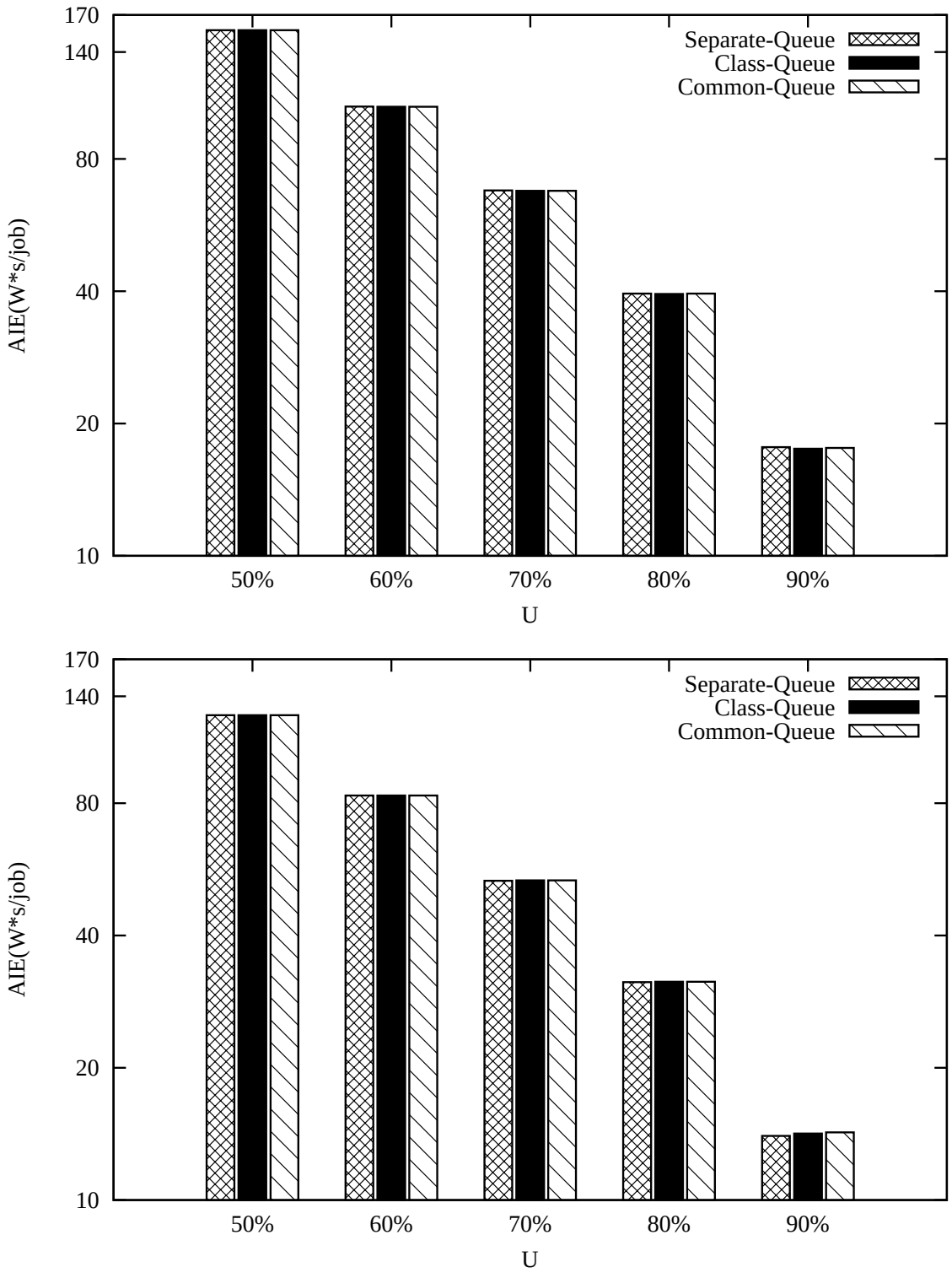


Figure 4: AIE vs. system load

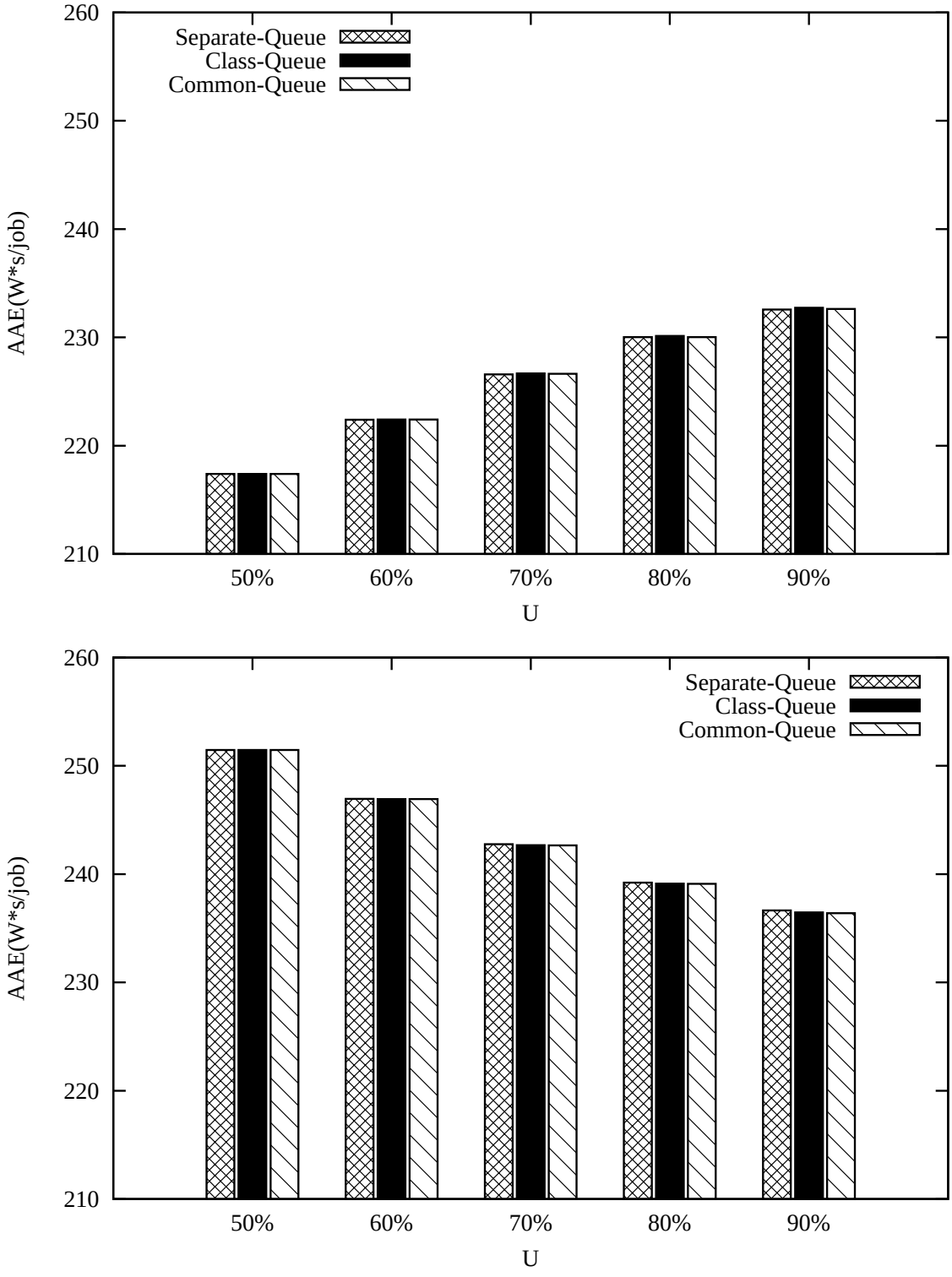


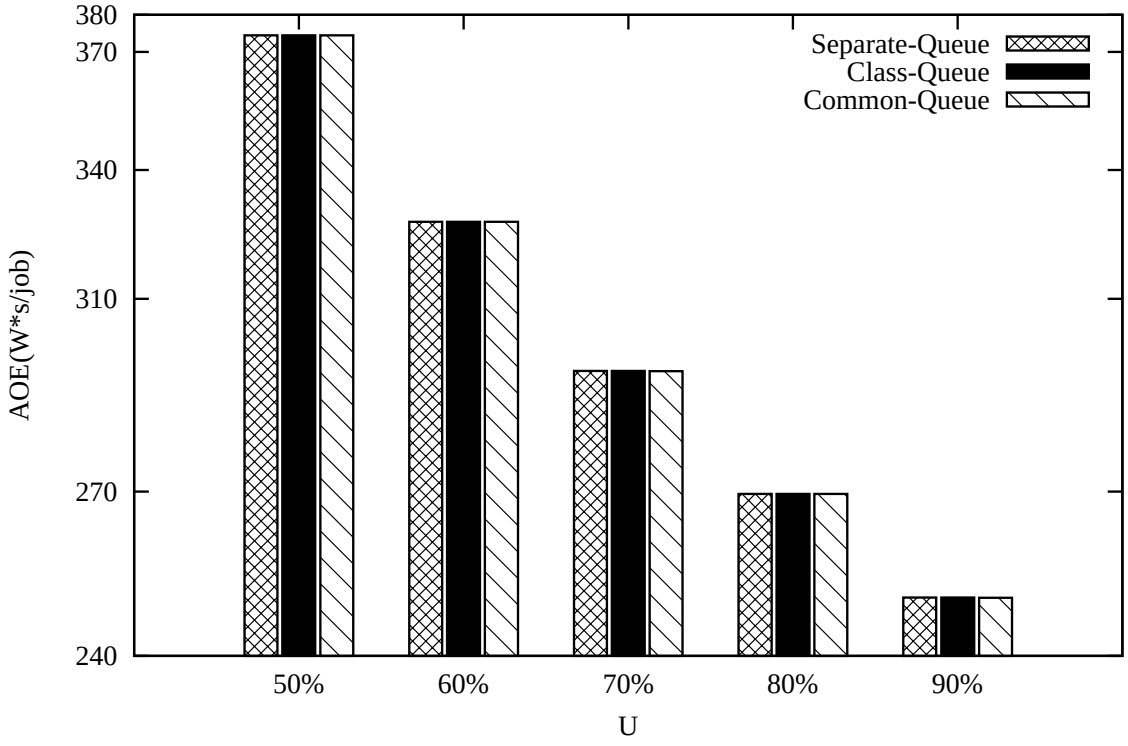
Figure 5: AAE vs. system load

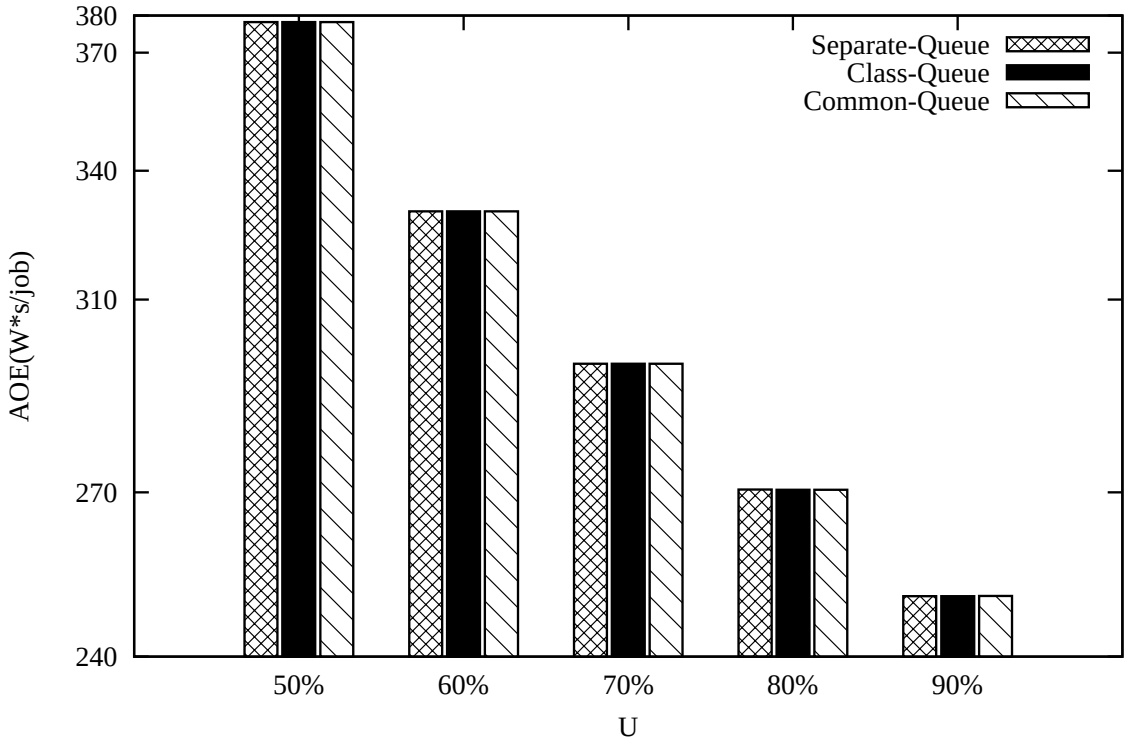
the smaller amount of idle energy they consume.

In the case of average active energy, the two policies show opposing tendencies: surprisingly, with HP priority, the AAE decreases as system load grows, while AAE increases with EE priority (Figure 5). The phenomenon can be explained by the trade-off between performance and energy saving. At medium system load, most job requests are executed on servers operating with lower energy cost, thus the energy dedicated to the execution of jobs is significantly smaller (217 $W.s$/job vs. 251 $W.s$/job). As demand for jobs grows, the use of high performance servers becomes inevitable, since the policies are based on shortest queue. In consequence, the difference in AAE between SQEE and SQHP decreases at high system load (232 $W.s$/job vs 236 $W.s$/job at 90% of utilization.

However, as it can be seen on Figure **??**, the average operating energies are almost the same in both policies, with the greatest difference noticeable at medium system load, where EE consumes 374 $W.s$/job and HP 378 $W.s$/job.

In comparison of the three queueing schemes, it can be declared that the energy use is nearly identical, independently of the applied policy. The less number of queues we have, the better each of them is utilized.


380
370
340
310
270
240
AOE(W*s/job)
Separate-Queue
Class-Queue
Common-Queue
50%
60%
70%
80%
90%
U


380
370
340
310
270
240
AOE(W*s/job)
Separate-Queue
Class-Queue
Common-Queue
50%
60%
70%
80%
90%
U

5.2.2 Performance

The performance is firstly measured in terms of service time, that is, the shorter the mean service time, the better a system performs. Figure 6 illustrates the mean service time against system load for all system models and policies.

The results show that tendencies observed in case of EE and HP priority are opposing: system models under the EE policy yield a shorter service time as system loads increase, while under the HP policy they give a longer service time. Although they all converge to the interval $(1.16, 1.18)$. This can be explained by the dominant utilization of high performing servers in case of high system loads under EE policy, and generally under HP policy. A better performing server requires a shorter amount of time to process the exact job as a lower performing, but less energy demanding server. The anticipation that the mean service times are not impacted by queueing schemes is confirmed by Figure 6.

The second parameter representing the performance is the mean response time (Figure 7). It can be seen that the proposed models outperform the traditional model under both policies and in every system load. In terms of response time, the Common Queue performs the best, especially under high system utilization. The difference in performance increase directly with system utilization: while no significant difference can be

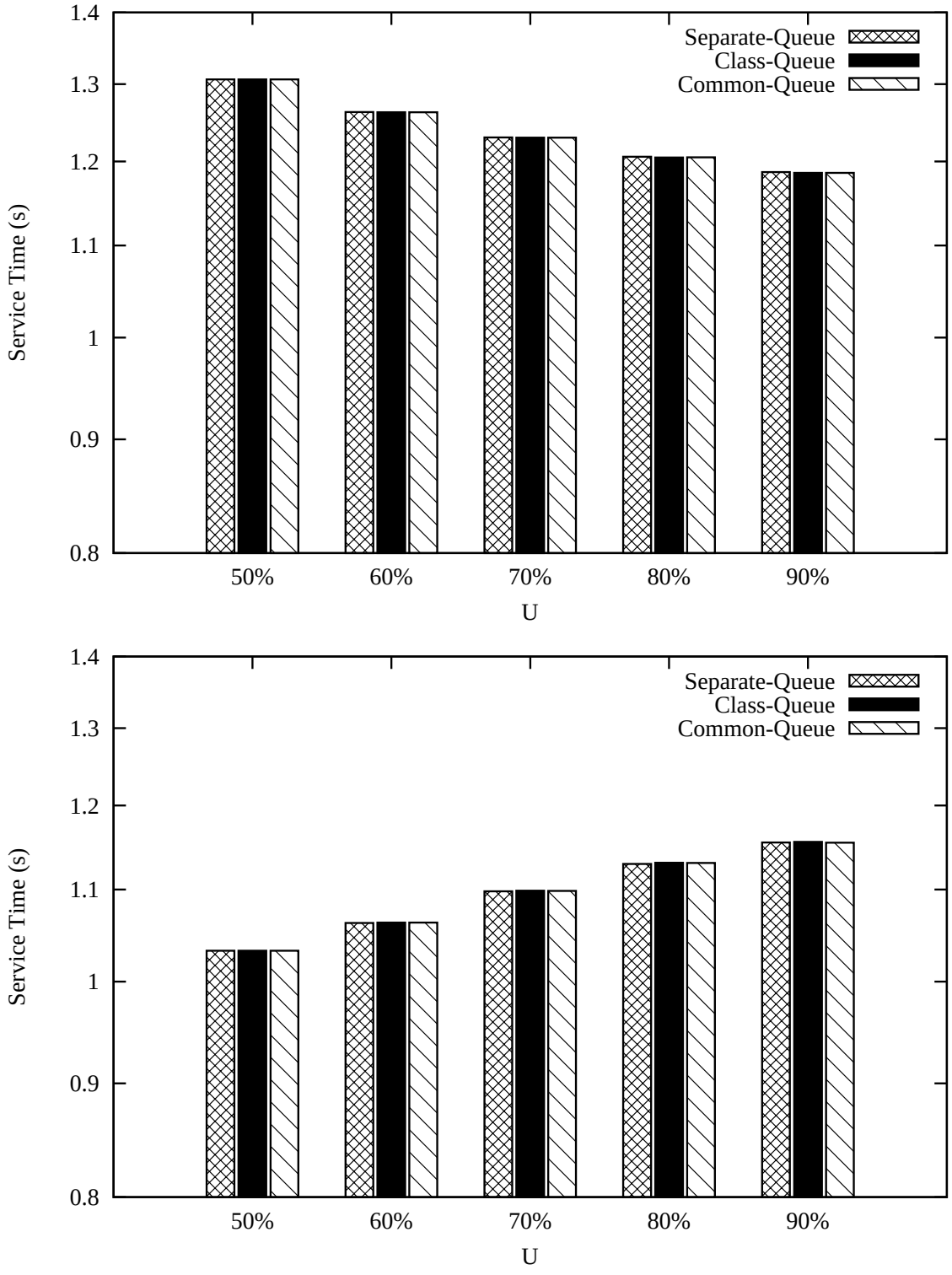


Figure 6: Mean service time vs. system load

discerned at 50%, a noticeable gap appears at 90%.

Lastly, the mean waiting time is compared in the scenarios (Figure 8).

It can be inferred from the bar charts that the expected waiting time of a job waiting in a queue is highest in the Separate Queue model, significantly lower in the Class Queue model, and slightly even lower in the Common Queue model. This is the result of the efficient use of local queues. By reducing the number of local queues and increasing the number of servers assigned to a queue, the event of a server becoming idle is taken advantage of more quickly. In conclusion, the figures tell us that the proposed system architecture models outperform the traditional architecture without additional energy consumption and energy cost.

6 Conclusion

In this paper we presented a generalized model for the performance evaluation of scheduling compute-intensive jobs with unknown service times in computational clusters. We gave an implementation of this model on three schemes (Separate Queue, Class Queue and Common Queue), differing in the way incoming jobs are buffered, and on two job-scheduling policies (SQEE, SQHP), prioritizing either energy efficiency or

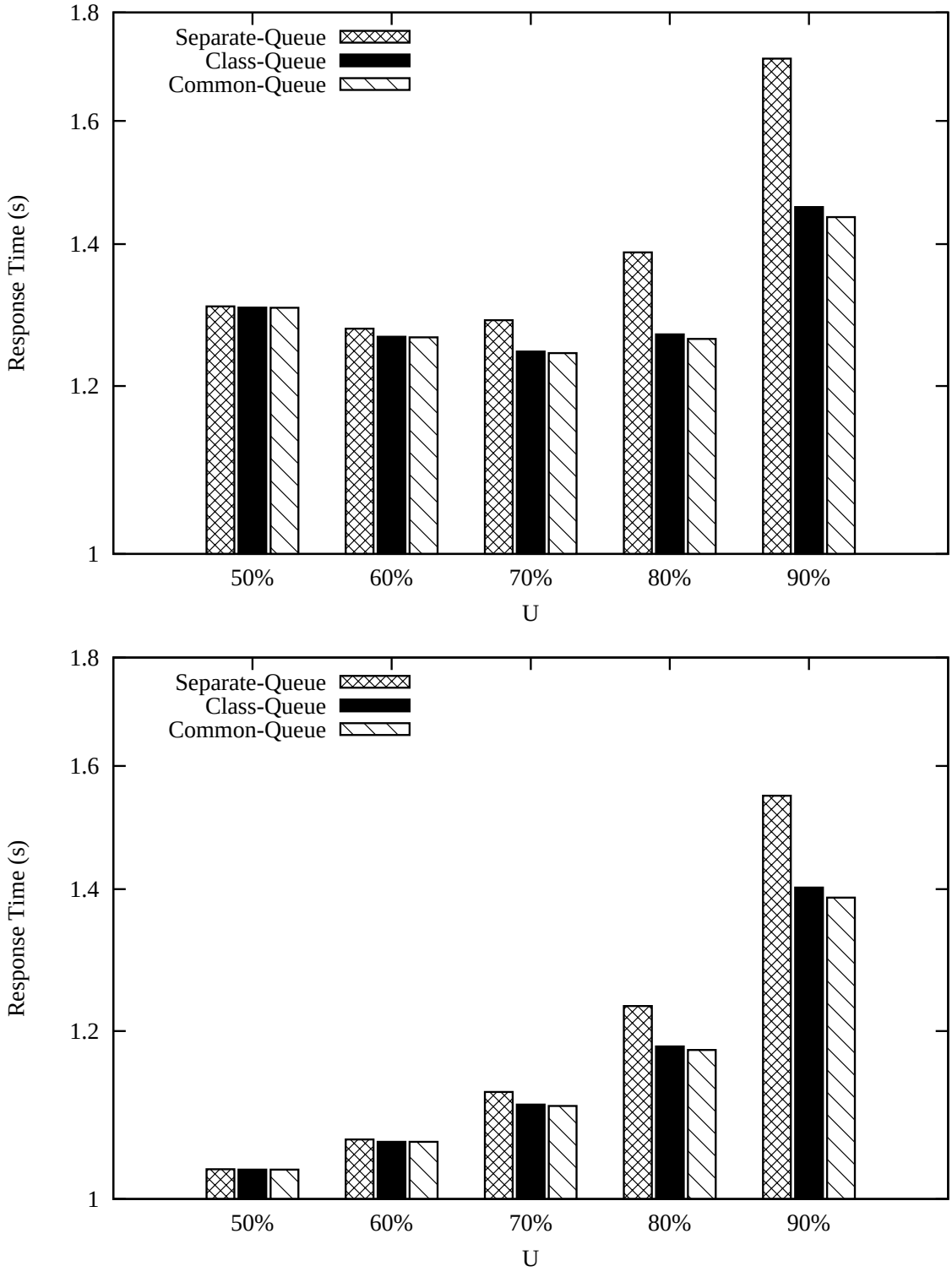


Figure 7: Mean response time vs. system load

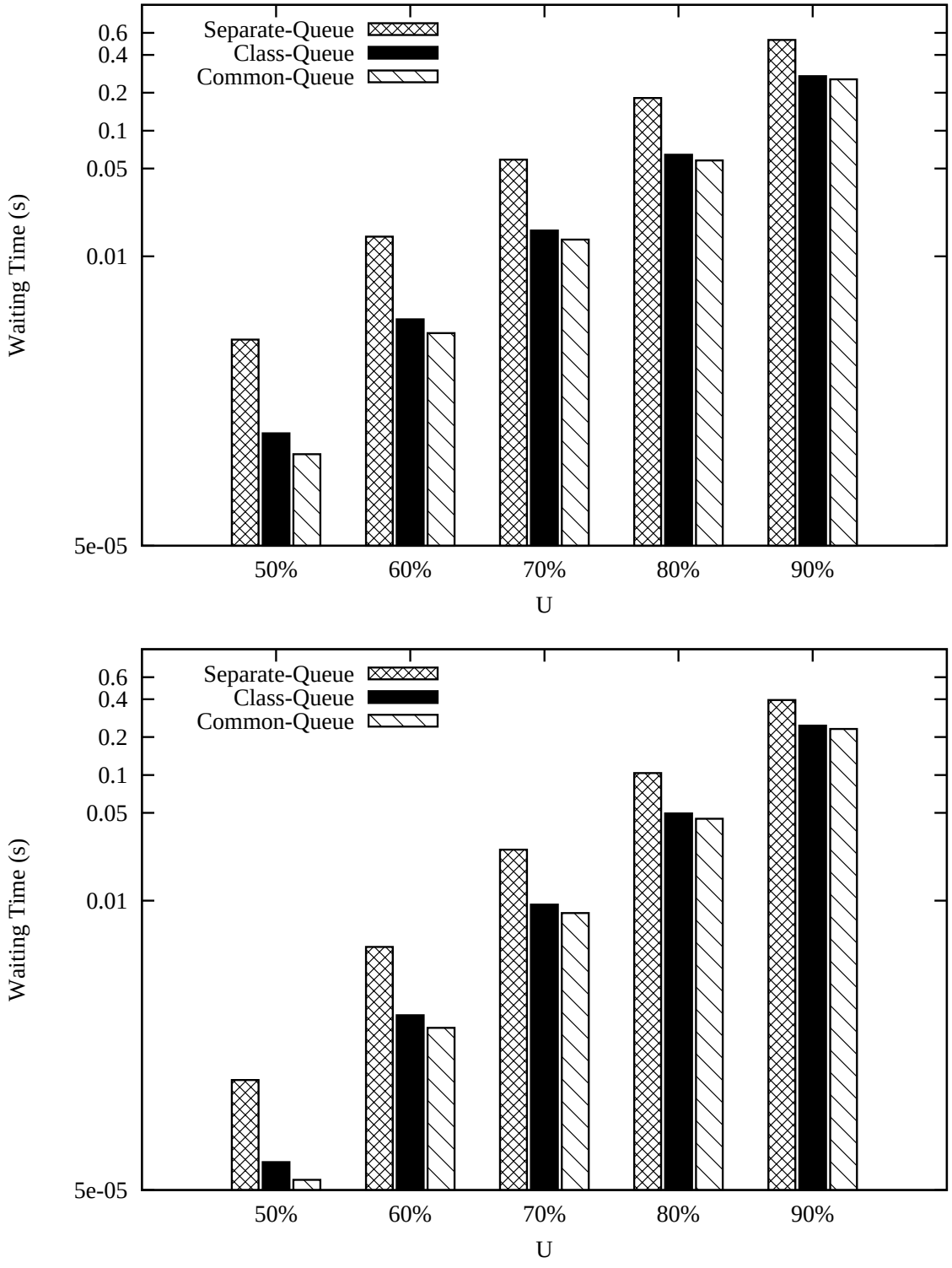


Figure 8: Mean waiting time vs. system load

high performance. We defined a ranking methodology of physical servers, fit to measure the performance and energy consumption of a system architecture.

Numerical results show that the buffering schemes do not affect the energy consumption of the investigated clusters. However, they have a significant impact on the mean response time and mean waiting time of incoming jobs, with Common Queue and Class Queue markedly outperforming Separate Queue. The gap between response- and waiting times directly increase with system load. Therefore the alteration of arriving job requests buffering can result in improved overall cluster performance without increased power consumption and energy cost.

References

[1] I. Foster, "The anatomy of the grid: enabling scalable virtual organizations," *The International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[2] B. Yagoubi and Y. Slimani, "Dynamic load balancing strategy for grid computing," *Transactions on Engineering, Computing and Technology*, vol. 13, pp. 260–265, 2006.

[3] G. Terzopoulos and H. D. Karatza, "Performance evaluation of a real-time grid system using power-saving capable processors," *The Journal of Supercomputing*, vol. 61, no. 3, pp. 1135–1153, 2012.

[4] A. Tchernykh, J. M. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, and S. Zhuk, "Two level job-scheduling strategies for a computational grid," in *Proceedings of the 6th international conference on Parallel Processing and Applied Mathematics*, PPAM'05, (Berlin, Heidelberg), pp. 774–781, Springer-Verlag, 2006.

[5] S. Zikos and H. D. Karatza, "Resource allocation strategies in a 2-level hierarchical grid system," *Simulation Symposium, Annual*, vol. 0, pp. 157–164, 2008.

[6] S. Zikos and H. D. Karatza, "A clairvoyant site allocation policy based on service demands of jobs in a computational grid," *Simulation Modelling Practice and Theory*, vol. 19, no. 6, pp. 1465–1478, 2011.

[7] S. Zikos and H. D. Karatza, "The impact of service demand variability on resource allocation strategies in a grid system," *ACM Trans. Model. Comput. Simul.*, vol. 20, pp. 19:1–19:29, Nov. 2010.

[8] Y. He, W. Hsu, and C. Leiserson, "Provably efficient online non-clairvoyant adaptive scheduling," in *Parallel and Distributed Processing Symposium, 2007. IPDPS*

2007. IEEE International, pp. 1 –10, march 2007.

[9] S. Zikos and H. D. Karatza, "Communication cost effective scheduling policies of nonclairvoyant jobs with load balancing in a grid," *Journal of Systems and Software*, vol. 82, no. 12, pp. 2103–2116, 2009.

[10] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 8, pp. 299 –316, june 2000.

[11] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, OSDI '94, (Berkeley, CA, USA), USENIX Association, 1994.

[12] S. Zikos and H. D. Karatza, "Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 239–250, 2011.

[13] G. Terzopoulos and H. D. Karatza, "Performance evaluation of a real-time grid system using power-saving capable processors," *The Journal of Supercomputing*, vol. 61, no. 3, pp. 1135–1153, 2012.

[14] K. Gkoutioudi and H. D. Karatza, "Multi-criteria job scheduling in grid using an accelerated genetic algorithm," *J. Grid Comput.*, vol. 10, no. 2, pp. 311–323, 2012.

[15] P. Fishwick, "Simulation toolkit." `http://www.cs.sunysb.edu/~algorith/implement/simpack/implement.shtml`.

[16] Standard Performance Evaluation Corporation. `http://www.spec.org/`.

[17] SPEC, "ssj_ops." `http://www.spec.org/power/docs/SPECpower_ssj2008-Result_File_Fields.html#Ops`.

[18] R. Chakka. *Performance and Reliability Modelling of Computing Systems Using Spectral Expansion.* PhD thesis, University of Newcastle upon Tyne (Newcastle upon Tyne), 1995.

[19] S. Zikos and H. D. Karatza. Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times. *Simulation Modelling Practice and Theory*, 19(1):239–250, 2011.