CREAT. MATH. INFORM.
Volume **25** (2016), No. 2,
Pages 223 - 235

Online version at `https://creative-mathematics.cunbm.utcluj.ro/`
Print Edition: ISSN 1584 - 286X; Online Edition: ISSN 1843 - 441X
DOI: https://doi.org/10.37193/CMI.2016.02.14

Analysis of finite-source cluster networks

ÁDÁM TÓTH, TAMÁS BÉRCZES, ATTILA KUKI, BÉLA ALMÁSI, WOLFGANG SCHREINER, JINTING WANG and FANG WANG

ABSTRACT. Nowadays the distributed heterogeneous resources of networks, like the computational grid, start to have a greater part of interest so, the investigations of such systems are vital. Because of the more efficient utilization of the resources, the job scheduling becomes more challenging for the system administrators. The allocation of the arriving jobs has a great impact on the efficiency and the energy consumption of the system. In this paper, we present a finite source generalized model for the performance evaluation of scheduling compute-intensive jobs based on the infinite model of Tien Van Do. The available computers are classified into three groups. This classification is based on two aspects: high performance priority (HP) and energy efficiency priority (EE). We investigate three schemes (separate queue, class queue and common queue) for buffering the jobs in a computational cluster that is built from Commercial Off-The-Shelf (COTS) servers. Our main interest is to calculate performance measures and energy consumption of the system using the different buffering schemes and classifications.

1. INTRODUCTION

In the literature, job allocation algorithms are proposed to schedule arriving jobs in computational clusters (see [2], [3] and references therein). In addition, some algorithms are designed with regard to the knowledge about characteristics of jobs. These may belong to either clairvoyant [9], [10] or non-clairvoyant algorithms [8].

Apart from the effective scheduling, the energy consumption of such grid systems turns into a really crucial requirement due to the rapid increase of the size of the grid and the goal of a green network. The most common techniques of reducing energy consumption are related to the Dynamic Power Management used in runtime. It is practical to examine algorithms which offer the greatest performance while using as low amount of energy as possible.

Do [1] introduced an infinite generalized model for the performance evaluation of scheduling compute-intensive jobs with unknown service times in computational clusters. In this paper we used a finite model instead of the infinite one to make the queueing model more realistic.

We investigated three cases from the viewpoint of buffering the arriving jobs: Separate Queue, Class Queue and Common Queue. The available computers are classified into three groups. This classification has been made according two aspects: high performance priority (HP) and energy efficiency priority (EE).

Because of the fact the state space of the describing Markov chain is very large, it is rather difficult to calculate the system measures in the traditional way of writing down and solving the underlying steady-state equations. To simplify this procedure we used the software tool MOSEL-2 (Modeling, Specification and Evaluation Language), to formulate the model and to obtain the performance measures.

Received: 12.12.2015. In revised form: 26.01.2016. Accepted: 01.02.2016
2010 *Mathematics Subject Classification.* 68M10, 68M20.
Key words and phrases. *Cluster network, performance evaluation, finite-source queueing system, Buffering scheme.*
Corresponding author: Tamás Bérczes; berczes.tamas@inf.unideb.hu

The organization of rest of the paper is the following: Section 2 contains the corresponding queueing model with components to study the behavior of the sensor nodes and the derivation of the main steady-state performance measures. In Section 3, we present some numerical examples. We illustrate the effect of the arrival rate and the listening period on the mean response times in the queue and in the orbit, respectively. Finally, Section 4 concludes the paper.

2. SYSTEM MODEL

It is assumed that the demands come to the system from a finite number of sources, and each entity in the source generates jobs according to exponential distribution with parameter λ. So the maximum number of the incoming jobs can be at most $N \times \lambda$, where N denotes the number of sources. Service times, which define the required time for the servers to execute the jobs, are exponentially distributed and each server possesses μ_i rate in class i. The service rate of the whole system can be defined as follows:

$$\mu_{\text{system}} = \sum_{i=1}^{I} J_i \times \mu_i \tag{2.1}$$

where J_i denotes the number of the processors in class i. The system load, the enormous carried traffic by the system, can be written as the ratio of between the arriving and the service rate:

$$\rho_{\text{system}} = \frac{\lambda \times \overline{N}}{\mu_{\text{system}}} \tag{2.2}$$

Because the numbers of sources of the considered models are finite, the stationary distributions exist in every moment, which implies the system stability.
At our inspected models the following scheduling policies are taken into account: SQEE and SQHP. These policies can be described shortly as follows:

- *SQEE (Shortest Queue with Energy Efficiency priority)*: This policy chooses the server with the shortest queue in the system. If there are more than one servers with this property, the server with the least energy consumption will be chosen. SQEE can be demonstrated as it yields the most energy efficiency at the cost of high performance.
- *SQHP (Shortest Queue with High Performance priority)*: Similar to SQEE, SQHP find the server with the shortest queue first. If more than one servers can be found, the one with the greatest performance will be chosen. SQHP can be demonstrated as it outperforms the other policies but it consumes more energy simultaneously.

At a computational cluster every physical server will be assigned to a specific type of server. Let denote S the set of type of servers and $I = |S|$ the size of S. Let's regard the type of server $s, s \in S$, which can be characterized with parameters C_s, $P_{ac,s}$ és $P_{id,s}$:

- C_s: This is an ssj_ops value, which is defined as the number of completed operations during the measurement of the interval divided by the number of seconds on this predefined interval. This parameter typifies the throughput of the computational server.
- $P_{ac,s}$: This is the energy consumption of the server under full load according to the benchmark of the Standard Performance Evolution Corporation (SPEC), which can be found in the SPECpower_ssj2008.
- $P_{id,s}$: This is the energy consumption of the server under idle state.

It is presumed that when a server is busy, then it works at throughput of C_s with estimated energy consumption of $P_{ac,s}$. When a server is idle, it can be switched off to a zero power consumption state for energy preservation, or the main internal clock of the CPU can be stopped via software utilizing the amount of energy $P_{id,s}$ then. It is worth to mention that according to SPEC measurement standard, $C_s/P_{ac,s}$, $s \in S$ is the ratio of performance and energy of class s, which describes the energy efficiency of the calculated element. Higher ratio means more energy efficiency.

We define two ranking functions, which rest on high performance and energy efficiency. These can be described as:

$$r_p(s) = \frac{C_s}{\max\limits_{i \in s} C_i}, \quad s \in S \tag{2.3}$$

$$r_e(s) = \frac{\frac{C_s}{P_{ac,s}}}{\max\limits_{i \in s} \frac{C_i}{P_{ac,i}}}, \quad s \in S \tag{2.4}$$

It is surmised that the regarded computational cluster is built from three types of Commercial Off-The-Shelf (COTS) servers. These types are showed in Table 1.

Type of server	C_s	$P_{ac,*}(W)$	$C_s/P_{ac,*}$	$P_{id,*}(W)$
Acer AW2000h-Aw170h F2 (Intel Xeon E5-2670)[5]	6419263	1700	3776	364
Acer AW2000h-Aw170h F2 (Intel Xeon E5-2660)[4]	5286503	1275	4146	331
PowerEdge R820 (Intel Xeon E5-4650L)[6]	2790966	457	6102	108

TABLE 1

Computing capacity and energy consumption of the three types of servers are indicated in Table 1 which is based on the specification SPECpower_ssj2008 of SPEC. Note, that computation capacity with ssj_ops [7] (the number of completed operations per seconds) and the energy consumption are measured under 100% target load while $P_{id,*}$ is the energy consumption of the idle servers. Table 2 represents the computed order resultant from the two ranking functions ($r_e(s)$ and $r_p(s)$).

Type of server	**Classification according to performance**	**Classification according to energy efficiency**
Intel Xeon E5-2670	$r_p(1) = 1.0$	$r_e(1) \approx 0.64$
Intel Xeon E5-2660	$r_p(2) \approx 0.82$	$r_e(2) \approx 0.66$
Intel Xeon E5-4650L	$r_p(3) \approx 0.43$	$r_e(3) = 1.0$

TABLE 2

2.1. **Load.** We construct a system which comprises three classes, where each class contains three servers, so the total number of servers is 9. As it is mentioned before, the jobs come to the system from source of N components, as well as the jobs are generated according to Poisson distribution and the service of the jobs follows exponential distribution. Every job requires a computing capacity equivalent to 6419253 (ssj_ops) in average, which means the average service time of the jobs is one second if the jobs are routed to the type of Intel Xeon E5-2670 server.

Let μ_1, μ_2, μ_3 denote the intensity rates, if the jobs are allocated to an Intel Xeon E5-2670, Intel Xeon E5-2660 and Intel Xeon E5-4650L server, respectively. It follows that $\mu_1 = \frac{1}{s}, \mu_2 = \frac{0.82}{s}$, $\mu_3 = \frac{0.43}{s}$, which is exemplified in Table 2 well.

From the equations 2.1 and 2.2 we get the service rate of the whole system. The required average time for serving a job is $\mu_{system} = 6.75$.

2.2. **Separate Queue.** In case of the Separate Queue polices, every server has its own queue as it can be seen in Figure 1. The local scheduler distributes the incoming jobs to the servers. The scheduling algorithm chooses that server which has the minimum queue length. If there are more than one free server or servers of same queue length, then the scheduler will choose the server with the highest priority (the lowest indexed). As mentioned before, the servers are classified into two groups:

- High Performance: In this case the servers with the highest performance get the highest priority (the lowest index).
- Energy Efficiency: In this case the servers with the lowest power consumption get the highest priority (the lowest index).

Henceforth let c_{ij} denote the jth server in class i, and q_{ij} the queue which belongs to the jth server in class i. So, the value range of c_{ij} can be between 0 and 1, which shows that jth server is either busy or not in class i. The value range of q_{ij} can spread from 0 to $N - I * J$, which shows how many jobs are located at the queue of jth server in class i.

The state of the network at time t can be considered as a continuous Markov-chain with dimension $I \times J + I \times J$:

$$X(t) = (c_{11}(t); \dots ; c_{IJ}(t); q_{11}(t); \dots ; q_{IJ}(t))$$

The system's steady-state probabilities can be defined the following way:

$$P(c_{11}; \dots ; c_{IJ}; q_{11}; \dots ; q_{IJ}) = \lim_{t\to\infty} P((c_{11}(t) = c_{11}; \dots ; c_{IJ}(t) = c_{IJ}; \\ q_{11}(t) = q_{11}; \dots ; q_{IJ}(t)) = q_{IJ})$$

As soon as the steady-state probabilities are known, further important system's performance measures can be calculated (for the parameters see Table 3.):

- R_{ij} – The probability that the jth server is busy in class i:

$$R_{ij} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{ij}=1}^{1} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_{11}=0}^{N-I*J} \dots \sum_{q_{IJ}=0}^{N-I*J} P(c_{11}, \dots , c_{ij}, \dots , c_{IJ}; q_{11}, \dots , q_{IJ})$$

- L_{ij} – The probability that the jth server is free in class i:

$$L_{ij} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{ij}=0}^{0} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_{11}=0}^{N-I*J} \dots \sum_{q_{IJ}=0}^{N-I*J} P(c_{11}, \dots , c_{ij}, \dots , c_{IJ}; q_{11}, \dots , q_{IJ})$$

- *The mean of queue ij length*:

$$Q_{ij} = \sum_{c_{11}=0}^{1} \cdots \sum_{c_{IJ}=0}^{1} \sum_{q_{11}=0}^{N-I*J} \cdots \sum_{q_{ij}=0}^{N-I*J} \cdots \sum_{q_{IJ}=0}^{N-I*J} q_{ij} * P(c_{11},\ldots,c_{IJ};q_{11},\ldots,q_{ij},\ldots,q_{IJ})$$

- *The mean number of jobs in the queues*:

$$\overline{Q} = \sum_{i=0}^{I}\sum_{j=0}^{J} Q_{ij}$$

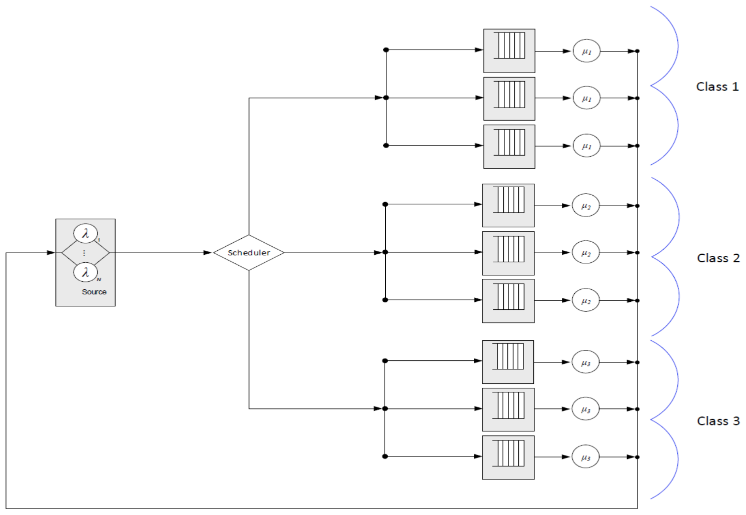

FIGURE 1. Model of the Separate Queue

2.3. **Class Queue.** In case of the Class Queue polices a common buffer is assigned to each class (see Figure 2). Jobs are scheduled to a specific class, and they will remain in the queue of that specific class until all the servers are busy. If a job departs from this class, then the first waiting job assigned to this class will be immediately routed to the free server. The algorithm controls the incoming jobs considering the following criteria: idle servers, priority of the classes (HP, EE), and the length of the queue.

Henceforth, let c_{ij} denote the jth server in class i, and q_i the queue which is assigned to class i. So the value range of c_{ij} can be between 0 and 1, which shows that jth server in class i is either busy or not. The value range of q_i can spread from 0 to $N - I * J$, which shows how many jobs are located at the queue of class i.

The state of the network at time t can be considered as a continuous Markov-chain with dimension $I \times J + I$:

$$X(t) = (c_{11}(t);\ldots;c_{IJ}(t);q_1(t);\ldots;q_I(t))$$

The system's steady-state probabilities can be defined the following way:

$$P(c_{11};\ldots;c_{IJ};q_1;\ldots;q_I) = \lim_{t\to\infty} P((c_{11}(t) = c_{11};\ldots;c_{IJ}(t) = c_{IJ};$$
$$q_1(t) = q_1;\ldots;q_I(t)) = q_I)$$

As soon as the steady-state probabilities are known, further important system's performance measures can be calculated (for the parameters see Table 3.):

- R_{ij} – The probability that the jth server is busy in class i:

$$R_{ij} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{ij}=1}^{1} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_1=0}^{N-I*J} \dots \sum_{q_I=0}^{N-I*J} P(c_{11},\dots,c_{ij},\dots,c_{IJ};q_1,\dots,q_I)$$

- L_{ij} – The probability that the jth server is free in class i:

$$L_{ij} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{ij}=0}^{0} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_1=0}^{N-I*J} \dots \sum_{q_I=0}^{N-I*J} P(c_{11},\dots,c_{ij},\dots,c_{IJ};q_1,\dots,q_I)$$

- *The mean length of queue i*:

$$Q_i = \sum_{c_{11}=0}^{1} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_1=0}^{N-I*J} \dots \sum_{q_i=0}^{N-I*J} \dots \sum_{q_I=0}^{N-I*J} q_i * P(c_{11},\dots,c_{IJ};q_1,\dots,q_i,\dots,q_I)$$

- *The mean number of jobs in the queues*:

$$\overline{Q} = \sum_{i=0}^{I} Q_i$$

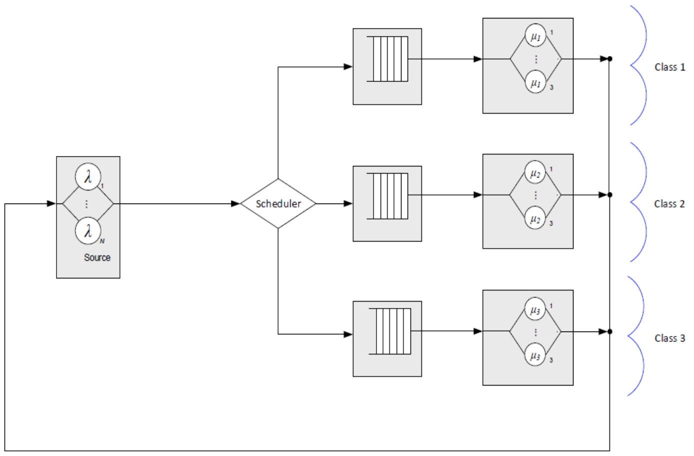

FIGURE 2. Class Queue model

2.4. **Common Queue.** In case of the Common Queue policies, only one buffer is available for all classes where the jobs can be waiting in (see Figure 3). If a job arrives then its service begins immediately if at least one server is idle. If more than one server is idle, then the local scheduler will choose the server with the highest priority (HP, EE). If all the servers are busy, then the local scheduler distributes the job into the queue and the job remains in the queue until one of the servers become idle.

Henceforth, let c_{ij} denote the jth server in class i, and let denote q_1 the single queue. So the value range of c_{ij} can be between 0 and 1, which shows that jth server in class *i* is either busy or not. The value range of q_1 can spread from 0 to $N - I * J$, which shows how many jobs are located at the queue.

The state of the network at time t can be considered as a continuous Markov-chain with dimension $I \times J + 1$:

$$X(t) = (c_{11}(t); \dots ; c_{IJ}(t); q_1(t))$$

The system's steady-state probabilities can be defined the following way:

$$P(c_{11}; \dots ; c_{IJ}; q_1) = \lim_{t \to \infty} P((c_{11}(t) = c_{11}; \dots ; c_{IJ}(t) = c_{IJ}; \\ q_1(t) = q_1)$$

As soon as the steady-state probabilities are known, further important system's performance measures can be calculated (for the parameters see Table 3.):

- R_{ij} – The probability that the jth server is busy in class i:

$$R_{ij} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{ij}=1}^{1} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_1=0}^{N-I*J} P(c_{11}, \dots , c_{ij}, \dots , c_{IJ}; q_1)$$

- L_{ij} – The probability that the jth server is free in class i:

$$L_{ij} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{ij}=0}^{0} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_1=0}^{N-I*J} P(c_{11}, \dots , c_{ij}, \dots , c_{IJ}; q_1)$$

- *The mean number of jobs in the queue*:

$$\overline{Q} = \sum_{c_{11}=0}^{1} \dots \sum_{c_{IJ}=0}^{1} \sum_{q_1=0}^{N-I*J} q_1 * P(c_{11}, \dots , c_{IJ}; q_1)$$

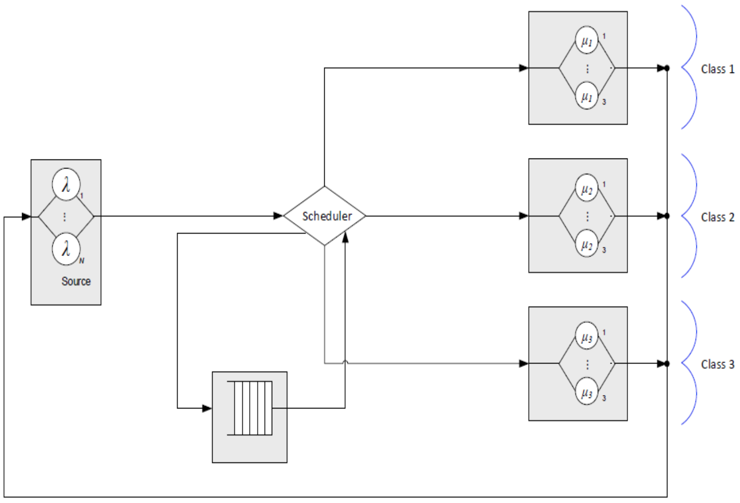

FIGURE 3. The model of the Common Queue

When all of the three buffer schemes are considered, the following further performance measures can be obtained by the help of the previously calculated performance characteristics:

- *The mean number of jobs at the servers*:

$$\overline{R} = \sum_{i=0}^{I} \sum_{j=0}^{J} R_{ij}$$

- *The mean number of jobs in the system*:

$$\overline{O} = \overline{Q} + \overline{R}$$

- *The mean number of jobs in the queue*:

$$\overline{N} = N - \overline{Q} - \overline{R}$$

- *The mean generating intensity*:

$$\overline{\lambda} = \lambda \overline{N}$$

- *The mean response time*

$$\overline{T} = \frac{\overline{O}}{\overline{\lambda}} :$$

- *The mean waiting time*

$$\overline{W} = \frac{\overline{Q}}{\overline{\lambda}} :$$

2.5. **Energy metrics.** Let $P_{id,i}$ and $P_{ac,i}$ denote the energy consumption of the idle and busy server, such that the energy consumption of the whole system can be defined the following way. In the first case the idle servers are not switched off (but in idle periods they consume some energy). The mean energy consumption of the system when the idle servers are not switched off:

$$AE_{no-switch} = \sum_{i=1}^{I} P_{ac,i} \sum_{j=1}^{J} R_{i,j} + P_{id,i} \sum_{j=1}^{J} L_{i,j} \tag{2.5}$$

The mean energy consumption of the system when the idle servers are switched off:

$$AE_{switch-off} = \sum_{i=1}^{I} P_{ac,i} \sum_{j=1}^{J} R_{i,j} \tag{2.6}$$

where $R_{i,j}$ is the probability that (i, j)-th server is busy and $L_{i,j}$ is the probability that (i, j)-th server is idle.

Notation	Parameter	Value
I	The number of server classes	3
J	Number of servers in the classes	3
μ_i	service rates of servers in class i	1; 0,82; 0,43
λ	jobs are generated according to this parameterized distribution	0,01-0,61
U	The mean system utilization	
N	Possible maximum numbers of jobs in the source	36
$P_a c_i$	Energy consumption of the server under busy state	see Table 1
$P_i d_i$	Energy consumption of the server under idle state	see Table 1

TABLE 3. Parameters

3. NUMERICAL RESULTS

In the previous section the construction of the introduced models was performed with the help of the MOSEL-2 program package and the system performance measures have been also calculated with the help of MOSEL-2. As mentioned before, we used three computational clusters and each cluster contains three servers and the different servers have different processors: Intel Xeon E5-2670, Intel Xeon E5-2660, and Intel Xeon E5-4650L. MOSEL-2 is a describing language which enables different program packages to use such as the SPNP (Stochastic Petri Net Package) or the TimeNet. The results created by MOSEL-2 can be illustrated graphically by the help of the IGL (Intermediate Graphical Language), which is part of MOSEL-2.

To evaluate the performance of the system the mean service, response and waiting times need to be observed and analyzed. Figure 4 show the mean service times in function of the arriving intensity for all system models.

At high priority mode, as the generating intensity increases, the mean service time also increases, while at energy efficiency mode it decreases. This phenomenon can be explained by that jobs will be scheduled to the servers which consume the least energy and these servers at the same time have the least computing capacity at EE. As the generating intensity increases, the computer having more computing capacity will be loaded, so the required time for execution of the jobs will be shorter in average in that way. A reverse process is true for HP, because first the jobs will be scheduled to the best servers in terms of performance. When it can be ascertained that each server is fully loaded, then the mean service time converges a constant value, around 1.33. Furthermore, it can be observed that for every queuing system the mean service times are almost the same, independent of the loads of the servers. The reason of this is that the same types and structures of servers are used in all three models.

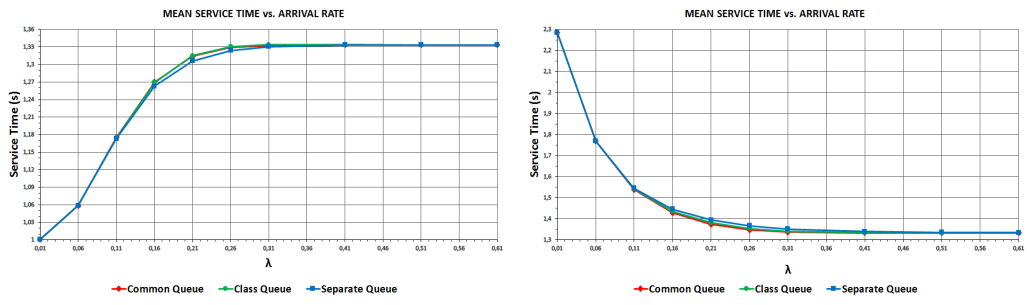

FIGURE 4. The mean service time applying HP the EE policy.

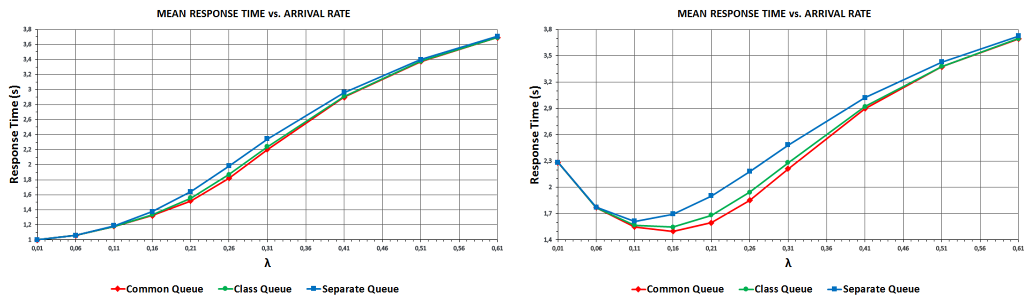

FIGURE 5. The mean response time applying HP the EE policy.

The mean response time at HP and EE policy is depicted in Figure 5. It can be seen that Class and Common Queue outperform Separate Queue policy until the system reaches the full load. It is clearly visible that Common Queue performs the best both at HP and EE policy. As long as the generating intensity does not reach 0.11, major differences can not be observed between the applied queuing systems. But in the range of 0.11 and 0.41 the difference appears vigorously, especially between the Separate Queue and the others. Moreover, an interesting effect can be experienced at EE policy namely the decreasing mean response time at first despite the increasing generating rate. This is explained by the fact that first the slowest servers will be loaded which will resulting high value of the mean response time. As the arrival rate starts to increase, quicker servers start to play more and more important role. Hereby the execution of the jobs become hastier, thus jobs spend less time in the system. Of course, this is true only for a specific intensity, because the more jobs arrive in the system, the more the system load is. This eventually means that every server will be busy, consequently jobs spend more time in the queue, too. In this case the mean response time will increase.

3.1. **Energy consumption.** In Figures 6 and 7 the mean energy consumption can be seen in case of switching off ($AE_{switch-off}$) and not switching off ($AE_{no-switch}$) the idle servers beside the HP and EE policies.

As we can see, there are not noticeable difference between buffering schemes using the HP policy. In case of the EE policy there, is little influence on the energy consumption. We get the lowest energy consumptions using separate queues and the highest using a common queue.

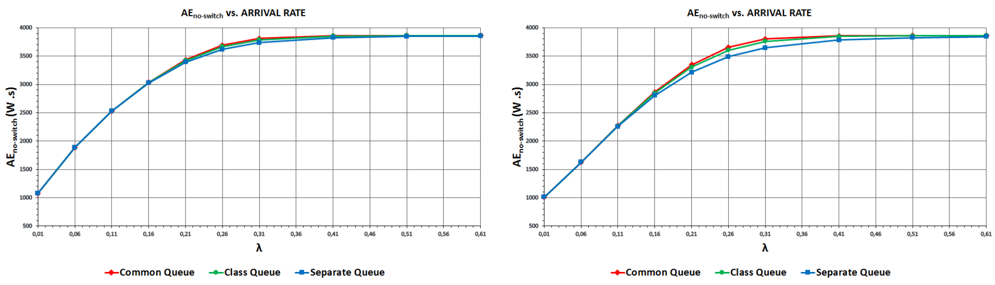

FIGURE 6. The mean energy consumption of the system using the HP and EE policy when idle servers are not switched off .

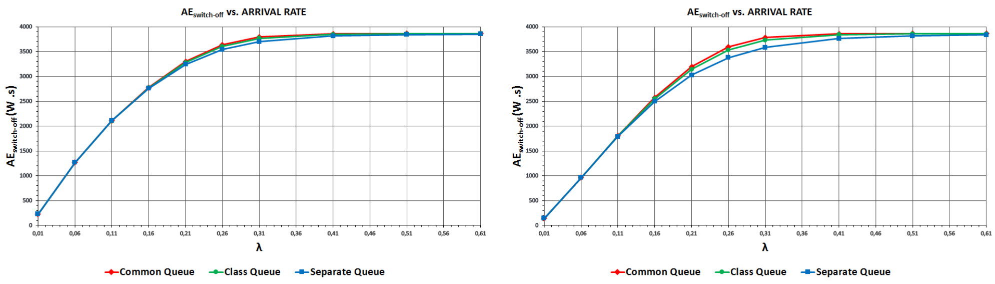

FIGURE 7. The mean energy consumption of the system using the HP and EE policy when idle servers are switched off .

Increasing the generating intensity the difference starts to disappear between the policies and finally both converge around 3800 W.s with an almost full system load. This can be explained by the fact that, if the probability of servers being busy is nearly one, then almost all the servers are working both at the HP and EE policies, so in this way energy saving is not achievable.

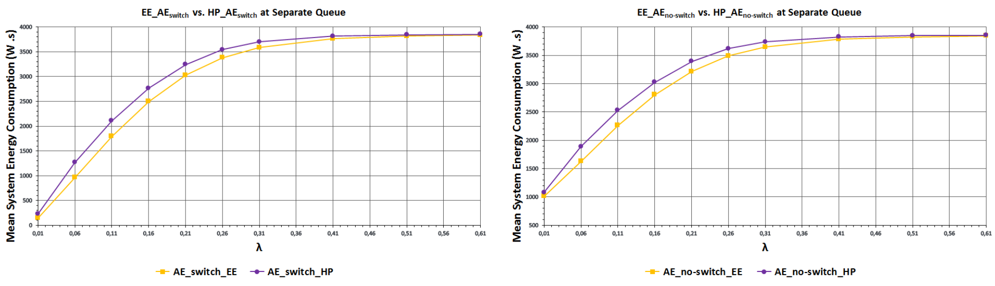

FIGURE 8. The mean energy consumption of the system using the HP and EE policies in case of switching and not switching off the idle servers at Separate Queue

.

Figures 8 represent the difference when the idle servers are not switched off and are switched off at Separate Queue, respectively. As it is mentioned before, significant differences are showing up till the system becomes nearly fully loaded. At very high system load, the two policies behave almost identically.

4. Conclusion

Scheduling jobs with large number of computation was considered in this paper. A generalized finite-source model was introduced for the performance evaluation of the system with unknown service rates in the clusters. Three schemes - Separate Queue, Class Queue and Common Queue - were investigated. A ranking methodology of physical servers performing the job-scheduling was defined.

It was shown by numerical results that the buffering schemes have only a very small effect on the energy consumption of the investigated clusters. However, a significant impact(momentous/remarkable influence) can be observed on the mean waiting time and mean response time of incoming jobs. Furthermore, the Separate Queue scheme proved less effective than the Common Queue and the Class Queue schemes. Therefore, choosing a good (best) buffering scheme can improve the overall cluster performance without increased power consumption(consuming more energy).

Acknowledgements. The work of Tamás Bérczes and Attila Kuki was granted by Chinese-Hungarian Bilateral Cooperation in Science and Technology project TéT 12 CN-1-2012-0009 (Chinese Grant No. 2013-83-6-34).

The present work of Ádám Tóth and Wolfgang Schreiner was supported by the Austro-Hungarian Cooperation Grant No 99öu6, 2015.

The research work of Jinting Wang and Fang Wang was partially supported by the National Science Foundation of China (Grant Nos. 11171019 and 71390334), and the Program for New Century Excellent Talents in University (No. NCET-11-0568)

References

[1] Do, T. V., Vu, B. T., Tran, X. T. and Nguyen, A. P., *A generalized model for investigating scheduling schemes in computational clusters*, Simulation Modelling Practice and Theory, **37** (2013), 30–42

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

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

[4] The Standard Performance Evaluation Corporation. Acer incorporated acer aw2000h-aw170h f2 (intel xeon e5-2660). Weboldal: https://www.spec.org/power_ssj2008/results/res2012q4/power_ssj2008-20120918-00546.html.

[5] The Standard Performance Evaluation Corporation. Acer incorporated acer aw2000h-aw170h f2 (intel xeon e5-2670). Weboldal: https://www.spec.org/power_ssj2008/results/res2013q1/power_ssj2008-20121212-00590.html.

[6] The Standard Performance Evaluation Corporation. Dell inc. poweredge r820 (intel xeon e5-4650l). Weboldal: https://www.spec.org/power_ssj2008/results/res2012q4/power_ssj2008-20121113-00586.html.

[7] The Standard Performance Evaluation Corporation. Specpower_ssj2008 result file fields? Weboldal: https://www.spec.org/power/docs/SPECpower_ssj2008-Result_File_Fields.html.

[8] Zikos, S. and Karatza, H., *Communication cost effective scheduling policies of nonclairvoyant jobs with load balancing in a grid*, Journal of Systems and Software, **82** (2009), No. 12, 2103–2116

[9] Zikos, S. and Karatza, H., *The impact of service demand variability on resource allocation strategies in a grid system*, ACM Transactions on Modeling and Computer Simulation, 20:19:1–19:29, 2010

[10] Zikos, S and Karatza, H., *A clairvoyant site allocation policy based on service demands of jobs in a computational grid*, Simulation Modelling Practice and Theory, **19** (2011), No. 6, 1465–1478

UNIVERSITY OF DEBRECEN
FACULTY OF INFORMATICS
EGYETEM TÉR 1, PO. BOX 12, 4010 DEBRECEN, HUNGARY
Email address: `adamtoth102@gmail.com`
Email address: `berczes.tamas@inf.unideb.hu`
Email address: `kuki.attila@inf.unideb.hu`
Email address: `almasi.bela@inf.unideb.hu`

JOHANNES KEPLER UNIVERSITY
RESEARCH INSTITUTE FOR SYMBOLIC COMPUTATION (RISC)
LINZ, AUSTRIA
Email address: `wolfgang.schreiner@risc.jku.at`

BEIJING JIAOTONG UNIVERSITY
DEPARTMENT OF MATHEMATICS, SCHOOL OF SCIENCE
BEIJING 100044, CHINA
Email address: `jtwang@bjtu.edu.cn`