Computing and Informatics, Vol. 36, 2017, 307–330, doi: 10.4149/cai_2017_2_307

NEW ALGORITHMS FOR BALANCING ENERGY CONSUMPTION AND PERFORMANCE IN COMPUTATIONAL CLUSTERS

Xuan Thi Tran

Analysis, Design and Development of ICT systems (AddICT) Laboratory
Department of Networked Systems and Services
Budapest University of Technology and Economics
Magyar tudósok körútja 2., H-1117 Budapest, Hungary

Tien Van Do*

Division of Knowledge and System Engineering for ICT (KSE-ICT)
Faculty of Information Technology
Ton Duc Thang University, Ho Chi Minh City, Vietnam
&
Analysis, Design and Development of ICT systems (AddICT) Laboratory
Department of Networked Systems and Services
Budapest University of Technology and Economics
Magyar tudósok körútja 2., H-1117 Budapest, Hungary
e-mail: dovantien@tdt.edu.vn, do@hit.bme.hu

Binh Thai Vu

Department of Networked Systems and Services
Budapest University of Technology and Economics
Magyar tudósok krt 2, H-1117 Budapest, Hungary

Abstract. In this paper, we propose new real-time measurement-based scheduling algorithms to achieve a trade-off between the energy efficiency and the performance

* Corresponding author

capability of computational clusters. An investigation is performed using a specific scenario of computing clusters with realistic parameters. Numerical results show that a trade-off between the performance and the energy efficiency can be controlled by the proposed algorithms.

Keywords: Heterogeneous resources, computational cluster, common queue model, real-time load

Mathematics Subject Classification 2010: 37M05, 97R30

1 INTRODUCTION

Computational clusters of heterogeneous distributed resources can be easily established based on high speed networks and powerful CPUs [1, 2, 3]. In such environments job scheduling is a challenging issue because various factors (quality of service, performance, energy consumption, etc.) could be taken into account [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].

To take a full advantage of computational systems, heterogeneity and large-scale sharing of resources can be exploited. Tang et al. [15] proposed a hierarchical reliability-driven scheduling scheme that can consider both global and local application reliability in computational grids. The aim [15] is to meet the reliability requirement of applications with the provisioning of quality of service. Kumar et al. [17] presented a dynamic load balancing algorithm based on a resource type policy to distribute works among resources. The author stated that tasks are allocated to available processors with the fair amount of time with their dynamic load balancing algorithm, while the makespan and the execution cost are reduced as well. Zikos and Karatza [12] studied three policies: SQEE (Shortest Queue with Energy Efficiency priority), SQHP (Shortest Queue with High Performance priority), and PBP-SQ (Performance-Based Probabilistic – Shortest Queue). It was shown [12] that SQEE is the most energy efficient at the price of a low performance and SQHP outperforms SQEE and PBP-SQ in the term of quality of service but comes with the highest energy consumption. Do et al. [16] showed that buffering schemes (separate queue, class queue, and common queue) play an important role regarding the waiting time and the response time experienced by jobs. They demonstrated that the common queue scheme outperforms the considered scheduling algorithms without an impact on the energy consumption of systems.

It is observed that previous works do not consider the variation of load offered to clusters, which motivates our study in this paper. Namely, we propose two dynamic scheduling algorithms that take into account real-time load variations. Numerical results show that a trade-off between the performance and the energy efficiency can be achieved by the proposed algorithms.

The rest of paper is organized as follows. New scheduling algorithms are introduced in Section 2. The numerical results are presented in Section 3. Finally, Section 4 concludes the paper.

2 SYSTEM DESCRIPTION AND PROPOSED SCHEDULING ALGORITHMS

2.1 Notations and Assumptions

We consider a heterogeneous computational cluster illustrated in Figure 1. The cluster has a common queue to store waiting jobs when all servers are busy. We assume that the cluster is built from K server types (server classes), indexed from 1 to K. Let $M(i)$ be the number of servers of type i used in the cluster. In the cluster, physical servers are indexed by pair (i, j), $(i = 1 \ldots, K;\ j = 1, \ldots, M(i))$.

Let S denote the set of server types. According to the `SPECpower_ssj2008` benchmark of the Standard Performance Evaluation Corporation (SPEC) [18], server type s, $s \in S$, is characterized with the following parameters:

- C_s, the number of operations finished during the measurement interval divided by the length of the measurement interval,
- the average active power $P_{ac,s}$, and
- the power consumption of a server in the idle state $P_{id,s}$.

Thus, $C_s/P_{ac,s}$ is the performance to power ratio of class s, and characterizes the energy efficiency of a physical server of class s.

Following [16], the two ranking functions, $r_p(s)$ and $r_e(s)$, can be formulated as follows:

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

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

Using function (1) and function (2), server classes are ranked in lists called a performance based list (S_p) and an energy consumption based list (S_e), respectively. Note that in the ranking lists, server classes are arranged in the decreasing order of priority. In addition, in the list S_p, if there are two server types with the same `ssj_ops` value, the server type with a higher average active power gets a higher index.

Furthermore, it is assumed that incoming jobs

- have service demand unknown to the local scheduler,
- are atomic jobs, which cannot be divided into smaller tasks,

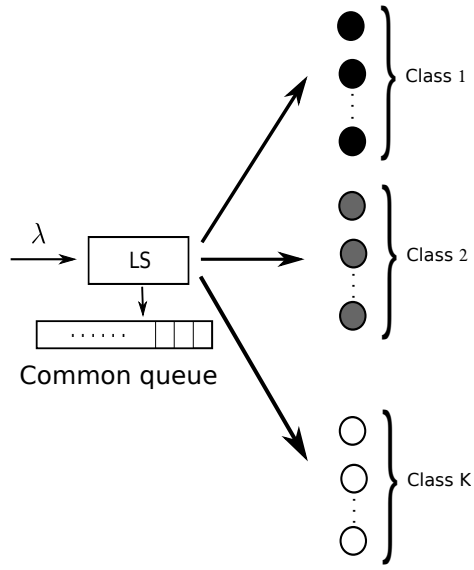

Figure 1. A common queue based cluster

- can be executed by any server,
- are served according to the First Come First Served (FCFS) service policy, and
- require a non-preemptible service.

2.2 Scheduling Algorithms

Let l and n be the number of incoming jobs and the number of completed jobs at the current time t. Let λ, μ and u denote system arrival rate, service rate, and load up to time t, respectively. These quantities are determined based on the historical data of the operation of the system until time t. The load is defined as the ratio between the arrival rate and the total service rate

$$u = \frac{\lambda}{\mu}. \tag{3}$$

Furthermore, we also measure a quasi-instantaneous load during durations of length TW (i.e., the time axis is divided to durations of length TW) as follows. In a current window w, let $a(w)$ be the number of arrivals. Then, the quasi-instantaneous arrival rate $\lambda(w)$ and the quasi-instantaneous load $u(w)$ can be calculated as

$$\lambda(w) = \frac{a(w)}{TW} \quad \text{and} \quad u(w) = \frac{\lambda(w)}{\mu}. \tag{4}$$

Let st_l be the time that takes to process job l. The average service time $ST(n)$ after the completion of n jobs is calculated as follows:

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

Let $m_k(n)$ be the number of completed jobs and $\theta_k(n)$ denote the sum of total service times of jobs at server class k until n jobs are completed. The weighted mean service time is defined as follows:

$$mean\_ST(n) = \frac{\sum_{k=1}^{K} M(k) \times \frac{\theta_k(n)}{m_k(n)}}{\sum_{k=1}^{K} M(k)}. \tag{5}$$

Our proposed algorithms are illustrated in Algorithm 1 and 2.

- Algorithm 1 evaluates the quasi-instantaneous load in each time window and compares with the average load. The idea is to apply the EE policy when the quasi-instantaneous load is lower than the average load and to choose the HP policy when the quasi-instantaneous load is higher than the average load.
- Algorithm 2 calculates average service time of n executed jobs ($ST(n)$) during run-time of system and compares with the overall mean service time per job from different resource classes ($mean\_ST(n)$). The aim is to apply the EE policy if the real-time mean service time is smaller, or switch to the HP policy if it is longer than the weighted average service time. The second proposal is called an Average Service Time (AVR-ST) algorithm.

Algorithm 1 Real Load algorithm

```
CALCULATE overall load u at the arrival of a new job,
CALCULATE quasi-instantaneous load of the current time window w, u(w)
if u(w) ≤ u then
    CHOOSE EE policy
else
    CHOOSE HP policy
end if
GOTO Algorithm 3
```

It is worth noticing that we initiate simulation run with EE policy until passing warm-up time of one thousand first jobs. Then the measures are used for policy determination.

2.3 Performance and Energy Metrics

To examine the impacts of our proposals, we use the same performance metrics as in [16]. Let wt_l be the waiting time of job l and rt_l be the system response time

Algorithm 2 Average Service Time algorithm

CALCULATE real-time average service time of historical jobs $ST(n)$ upon the arrival of a job
CALCULATE the overall mean service time $mean\_ST(n)$
if $ST(n) \leq mean\_ST(n)$ **then**
    CHOOSE EE policy
else
    CHOOSE HP policy
end if
GOTO Algorithm 3

Algorithm 3 Schedule

for (i, j) in *server_list* **do** ▷ *server_list* with priority order
    if server (i, j) is FREE **then**
        $free\_server \leftarrow (i, j)$
        GOTO ALLOCATE
    end if
end for
ALLOCATE:
if found *free_server* **then**
    ROUTE job to *free_server*
else
    RETAIN job in Common Queue
end if

to job l, then $rt_l = wt_l + st_l$. Mean waiting time $WT(n)$ and mean response time $RT(n)$ after completion of n jobs are calculated as follows:

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

and

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

The long-term averages of service times, waiting times, and response times are respectively defined as:

$$ST = \lim_{n \to \infty} ST(n),$$

$$WT = \lim_{n \to \infty} WT(n),$$

$$RT = \lim_{n \to \infty} RT(n).$$

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

The idle power and the active power consumption of server (i, j) are denoted by $P_{id,i}$ and $P_{ac,i}$, respectively. Server (i, j) consumes energy $P_{ac,i}$ when it processes jobs. However, even in the idle state if the server is not switched off, it still consumes energy $P_{id,i}$. Let $AAE(t)$ denote the mean energy consumption per job until t when n jobs are completed with the assumption of switching off idle servers and $AOE(t)$ denote the mean energy consumption per job until t in a case where servers stay in the idle state when not serving jobs.

The long-term average energy consumptions per job (AAE and AOE) are defined as

$$AAE = \lim_{n\to\infty} AAE(t), \tag{6}$$

$$AOE = \lim_{n\to\infty} AOE(t). \tag{7}$$

All presented parameters are summarized and listed in Table 1.

K	number of server classes
$M(i)$	number of servers in class i
μ	service rate of system up to time t
λ	system arrival rate up to time t
u	average system load up to time t
$u(w)$	measured load in time window w
$\lambda(w)$	measured arrival rate in w
$mean\_ST(n)$	mean service time per job divided among resource classes
$ST(n)$	measured average service time per job of n historical jobs
ST	the long-term average service time
WT	the long-term average waiting time
RT	the long-term average response time
AAE	average energy consumption per job with switching-off
AOE	average energy consumption per job with no switching-off

Table 1. Notations

3 NUMERICAL RESULTS

We built a simulation software which incorporates the cluster model presented in Section 2. We use the statistical module of the Telecommunication Networks Group, Politecnico di Torino[1] to collect simulation data and to perform the analysis of simulation runs. Note that the confident level of 99 % of simulation runs is applied

[1] `http://www.telematica.polito.it/oldsite/class/statistics.ps.gz`, Access September 19, 2014.

with the use of the statistical module. In most of cases, the accuracy (i.e. the ratio of the half-width of the confidence interval and the mean of collected observations) of all performance metrics is 0.005 at most.

3.1 The Parameters of a Computational Cluster

We consider a cluster that is built from three different types of Commercial Off-The-Shelf (COTS) servers, of which parameters are reported according to the specification `SPECpower_ssj2008` of SPEC [18]. Assume that considered computational cluster consists of twenty-four servers entirely, ($K = 3$, $M(i) = 8$, $i = 1, 2, 3$). Chosen server types and their parameters are presented in Table 2.

Server Type	C_s	$P_{ac,*}$ (W)	$C_s/P_{ac,*}$	$P_{id,*}$ (W)
Acer AW2000h-Aw170h F2 (Intel Xeon E5-2670) [19]	6 419 253	1 700	3 776	364
Acer AW2000h-AW170h F2 (Intel Xeon E5-2660) [20]	5 286 503	1 275	4 146	331
PowerEdge R820 (Intel Xeon E5-4650L) [21]	2 790 966	457	6 102	108

Table 2. Server specifications

The values of ranking functions are reported in Table 3.

Server Type	**Ranking Based on Performance**	**Ranking Based on 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 3. Ranking function

Each job requires a computing capacity equivalent to 6419253 (ssj_ops) in average, which means the average execution time of jobs is one second if jobs are routed to a server with Intel Xeon E5-2670 processor. The service rate of Intel Xeon E5-2670 is $1/s$, the service rate of Intel Xeon E5-2660 is $0.82/s$ and the service rate of Xeon E5-4650L is $0.43/s$ (if a job is executed by the corresponding CPU).

Our investigation is carried out with load U equal to 50 %, 60 %, 70 %, 80 % and 90 % (i.e., the mean inter-arrival time of jobs is 0.1111 s, 0.0926 s, 0.0794 s, 0.0694 s and 0.0617 s, respectively).

3.2 Job Balance

Figures 2–5 show the assignment ratio of jobs into three classes of servers (according to three processor types: E5-2670, E5-2660, and E5-4650L) when the cluster is

operated with the EE policy, the HP policy, the real load (RL) – and the average service time (AVR-ST) – algorithms, respectively.

When the EE policy and the HP policy are utilized, the ratio of jobs distributed into three classes is far different at the medium loads, but gets more equalized at high loads.

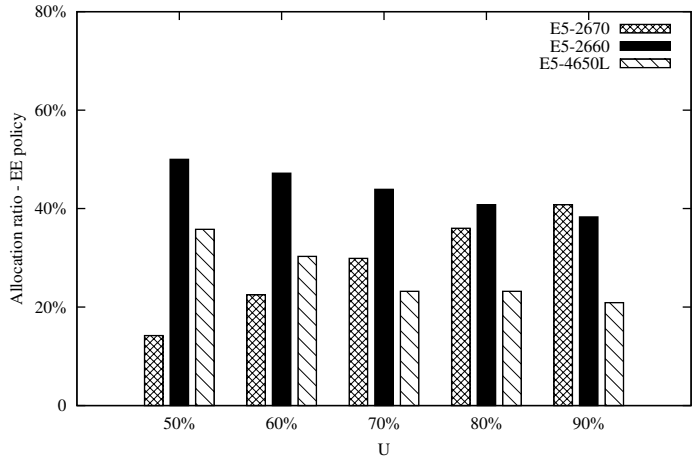

Figure 2. Job distribution vs. system load (EE policy)

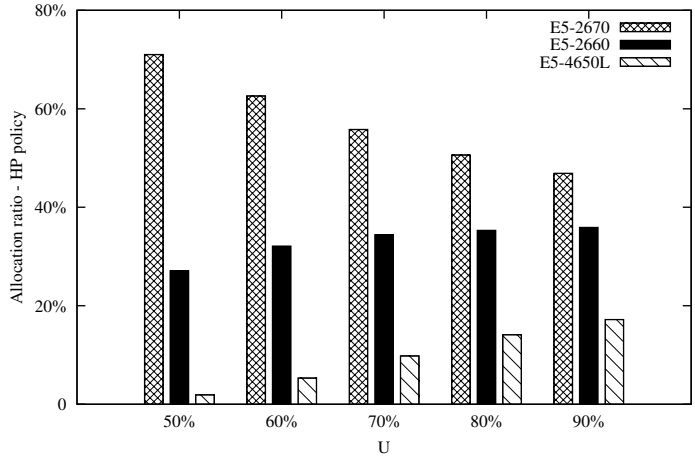

Figure 3. Job distribution vs. system load (HP policy)

When the EE policy is applied, the significant number of jobs are assigned to the lower performance classes at medium loads (see Figure 2). The class of E5-

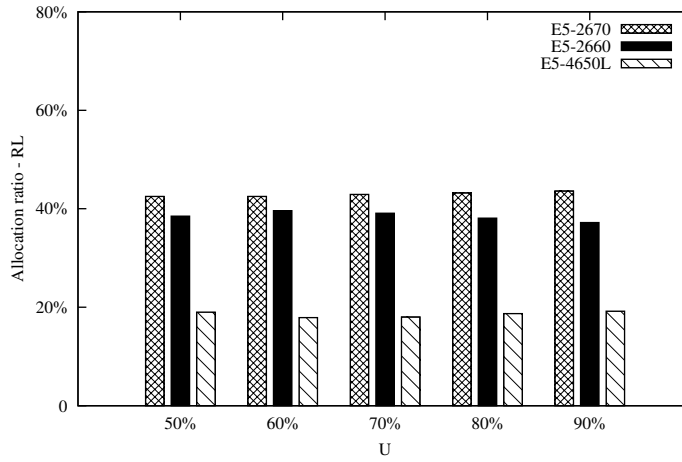

Figure 4. Job distribution vs. system load (RL)

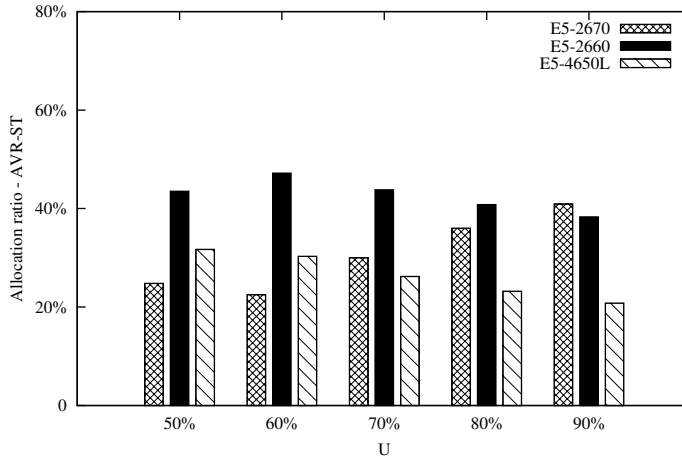

Figure 5. Job distribution vs. system load (AVR-ST)

2660 processors executes $\approx 50\,\%$, the E5-4650L processor class executes $\approx 36\,\%$ of jobs, but only around $14\,\%$ of jobs are assigned to the class of E5-2670 processors at $U = 50\,\%$. At higher loads more jobs are served by the E5-2670 class due to lower-performance classes are occupied.

On the contrary, when the HP policy is utilized, more jobs are executed by servers of type E5-2670 because of their high performance capacity. The ratio of jobs scheduled into this server class is far higher than that at class of the lowest – performance processors (E5-4650L).

When the RL algorithm is applied, job distribution remains unchanged in regard to the system load. Specifically, about 43 % of jobs are executed by E5-2760, 38 % by E5-2660, and 19 % by E5-4650L, see Figure 4. The AVR-ST algorithm also yields a better job distribution at 50 % of the system load, but the similar behavior at higher loads, in comparison to the EE policy. Furthermore, jobs are more evenly distributed into server classes by the new algorithms than by the EE and the HP policy.

3.3 Quality of Service Parameters

The mean service time, the mean waiting time and the mean response time per job are plotted in Figures 6–8, respectively. It can be seen from Figure 6 that the increase in the system load comes with the decrease in the mean service time when the EE policy is applied and the increase when the HP policy is used, while the mean service time is nearly unchanged when the RL algorithm is applied. Figure 7 plots the mean waiting time regarding to the system load. As expected, the mean waiting time is most favorable when the HP policy is applied, but the difference becomes negligible for high loads.

Figure 8 shows that the proposed algorithms result in a balance between features of the EE and the HP policy.

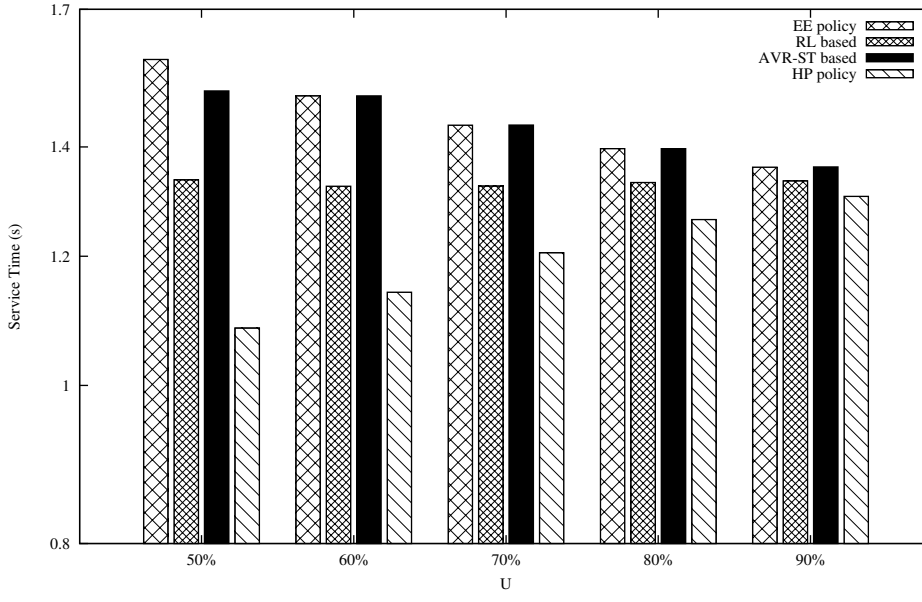

Figure 6. Mean service time vs. system load

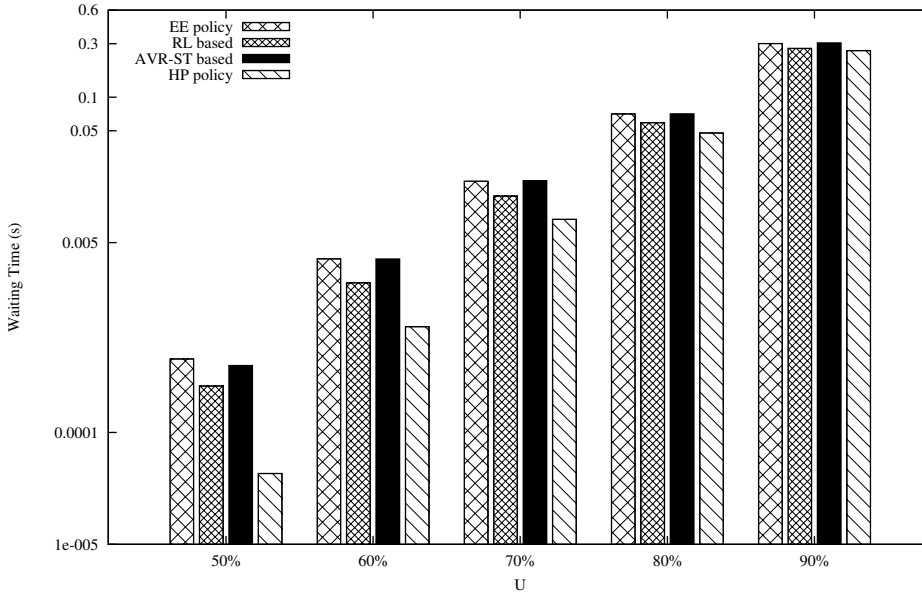

Figure 7. Mean waiting time vs. system load

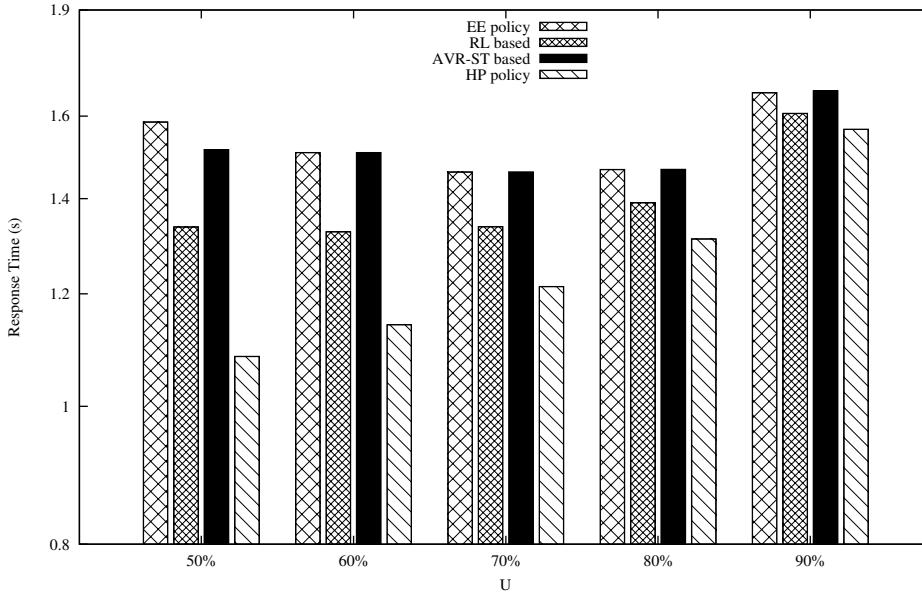

Figure 8. Mean response time vs. system load

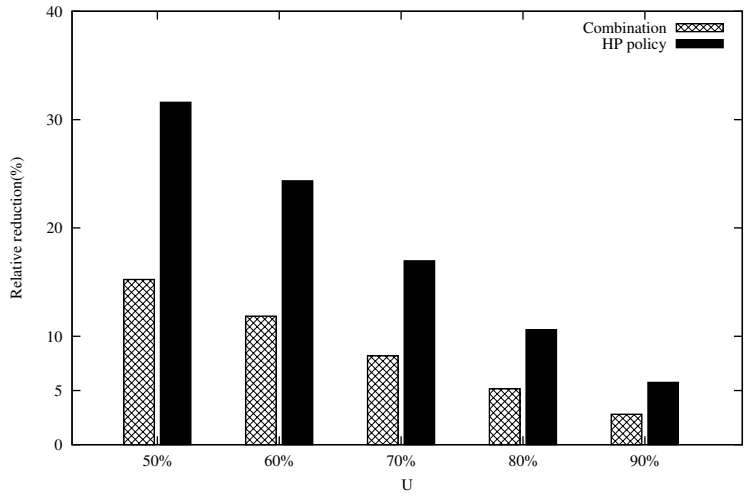

Figure 9. Relative reduction in response time vs. system load

In Figure 9, a relative reduction in the system response time is plotted when the RL algorithm or the HP policy is applied instead of the EE policy. We should note that there is no significant difference in the response time achieved by the EE policy and the AVR-ST algorithm. The results show that compared to the EE policy, the HP policy gives a reduction in percentage of 31.6 % to 5.74 %, while the RL algorithm achieves a reduction range from 15.24 % to 2.8 % when the system load increases from 50 % to 90 %.

3.4 Energy Consumption

The mean system energy consumption for serving one job, if idle servers are switched off or not switched off, are presented in Figures 10 and 11, respectively. It is worth emphasizing that when processors are idle, without switching them off, they still consume energy, but less due to the dynamic power management technique.

It is shown that with the increase in the system load, energy consumption is gradually increased under the EE policy or the AVR-ST, and decreased under the HP policy. It retains the same consumption when the RL algorithm is applied. The energy is saved most by the EE policy, medium by the new algorithms and least by the HP policy.

We plot the relative increase in the energy consumption when the HP policy or the RL algorithm are used instead of the EE policy in Figure 12. It is observed that the RL algorithm has a lower energy consumption than the HP policy.

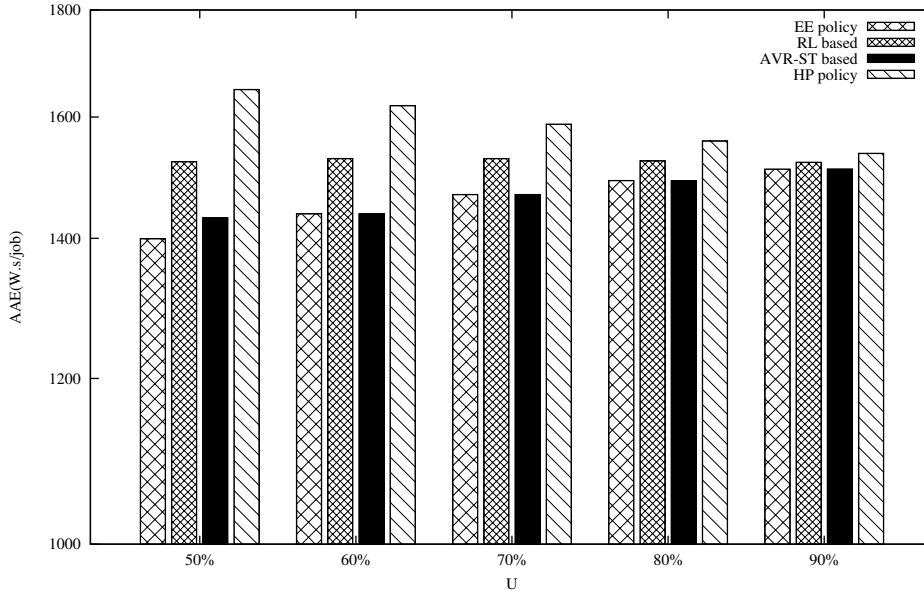

Figure 10. Energy consumption (switch-off) vs. system load

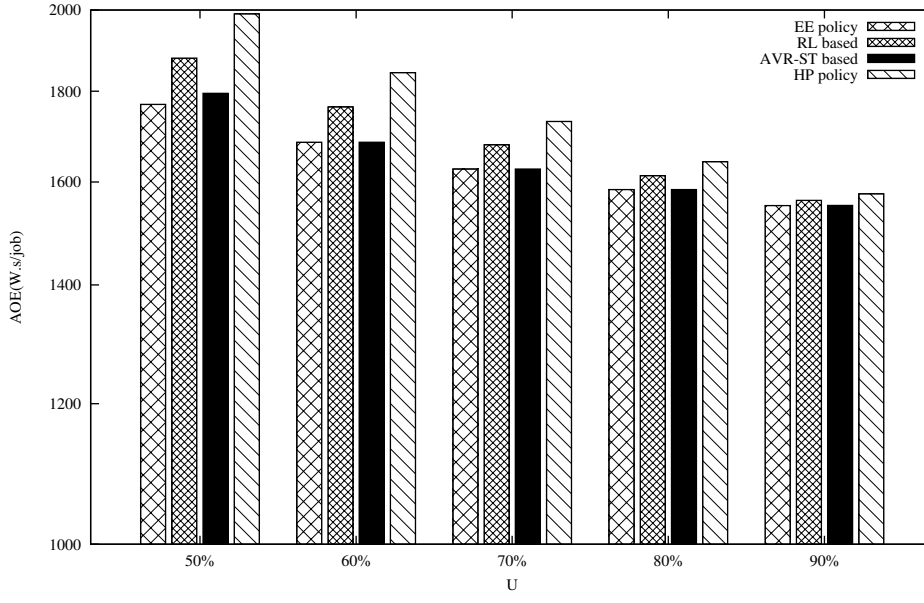

Figure 11. Energy consumption (no switch-off) vs. system load

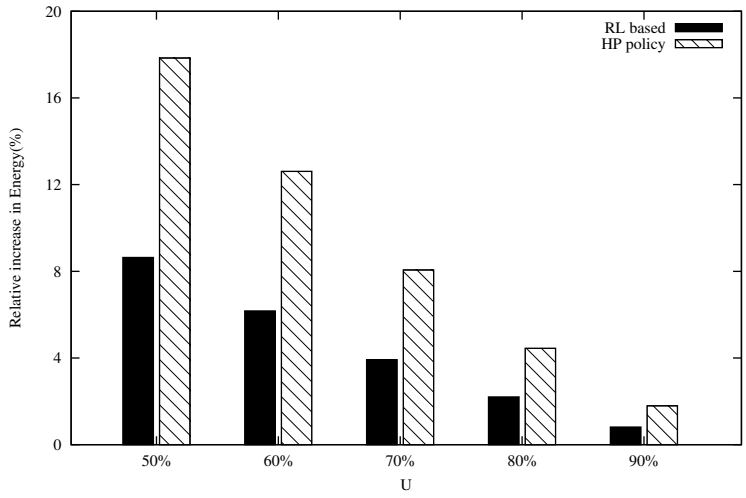

Figure 12. Relative increase in energy consumption vs. system load

3.5 Impacts of DVFS

We investigate the impact of DVFS technique that lowers the processing performance and the voltage of CPU, alongside with switching idle servers off, so that every processor handles jobs with the capacity of 70 % workload. The approximate values of the throughput and the average active power of processors after applying DVFS are showed in Table 4.

Server Type	C_s	$P_{ac,*}$ (W)
Intel Xeon E5-2670	4 517 449	1 169
Intel Xeon E5-2660	3 706 521	881
Intel Xeon E5-4650L	1 961 157	317

Table 4. Server specifications at 70 % workload

Figures 13 and 14 plot the system energy consumption and the response time where the system utilization is kept in the range of 50 % to 90 % (relatively to the reduced processing capacity). Compared to the system behaviors when processors run at full workload (see Figures 8 and 10), with DVFS, the system decreases around 30 % in performance, while it achieves only about 2 % of saving energy, regardless to algorithms studied.

In Figures 15 and 16, we show the relative degradation in the response time and the relative energy saving when DVFS is applied for $\lambda = 9.0$, and 10.8.

These results demonstrate that DVFS saves insignificant amount of energy regarding to HP policy and RL algorithm, and even badly affects energy efficiency in cases of the EE policy and the AVR-ST algorithm, where the efficiency fractions are

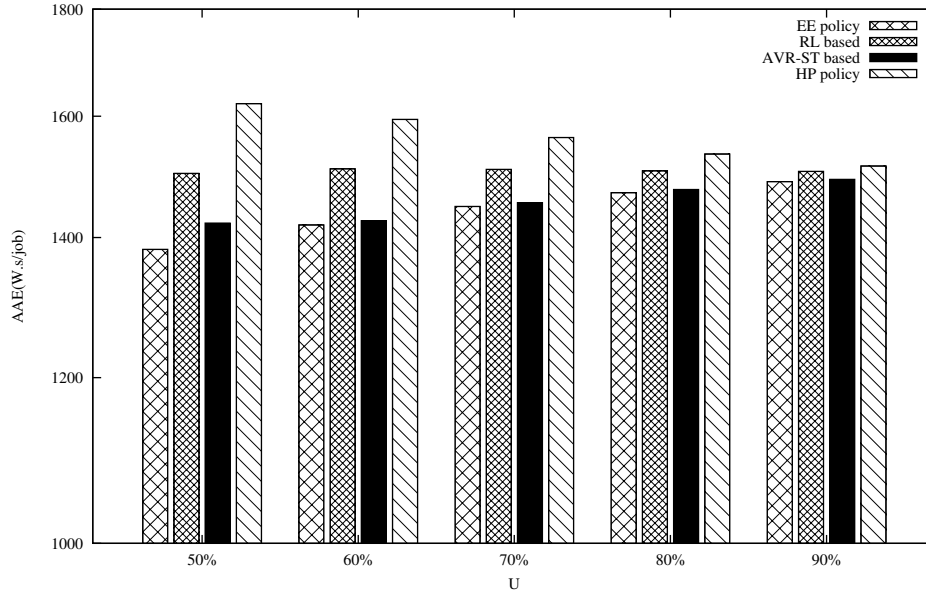

Figure 13. Energy consumption with DVFS applied vs. system load

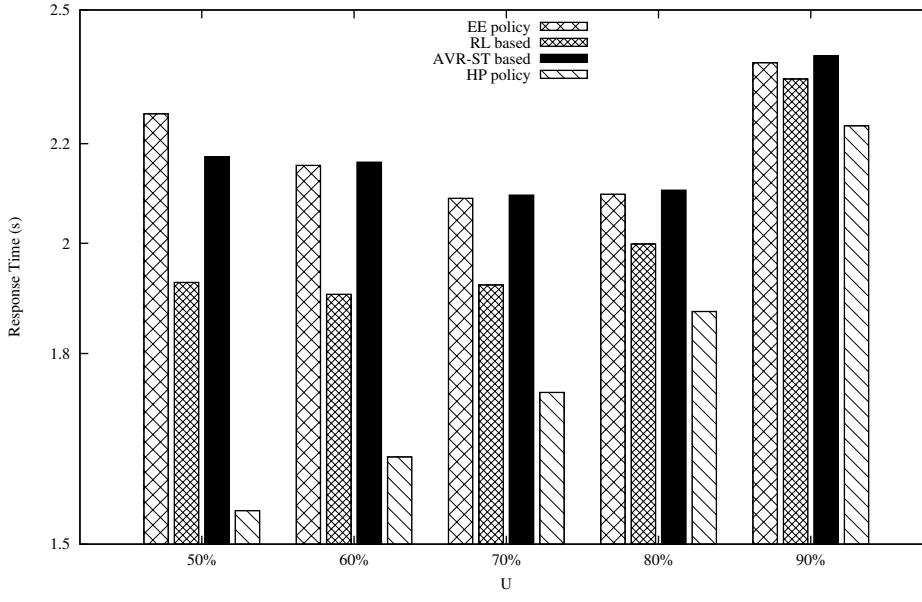

Figure 14. Mean response time with DVFS applied vs. system load

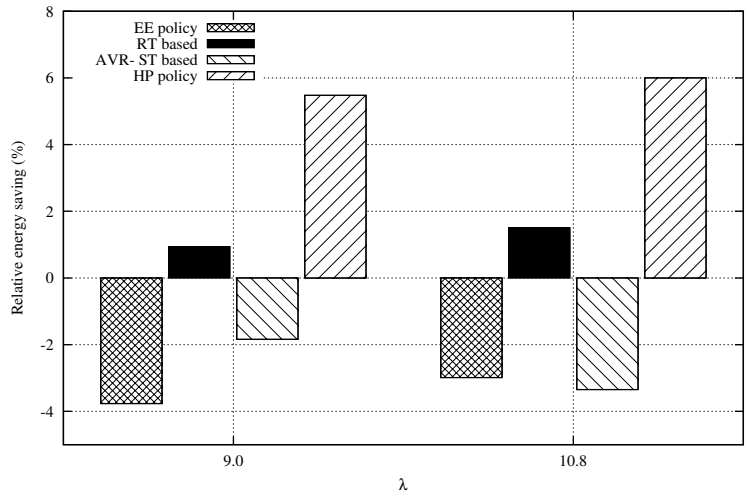

Figure 15. Energy saving ratio vs. λ

negative. Furthermore, it causes a quite dramatic degradation in response times of the system.

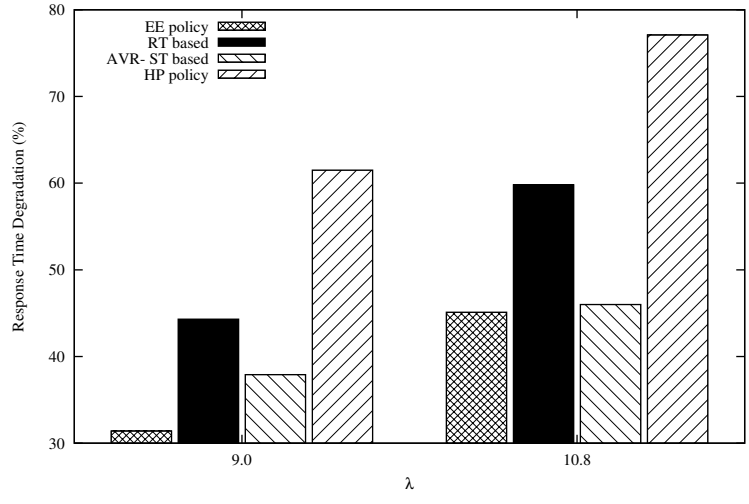

Figure 16. Degradation of response time vs. λ

3.6 Evaluations with Workload Traces as Input Data

In this section, we investigate the performance using the traces of jobs that were executed in a production environment of a specific company (two traces named INCC and UPR of one day were collected). From the quantile-quantile plots illustrated in Figures 17, 18, 19 and 20), it is obvious that the inter-arrival times and the service times of jobs do not follow an exponential distribution.

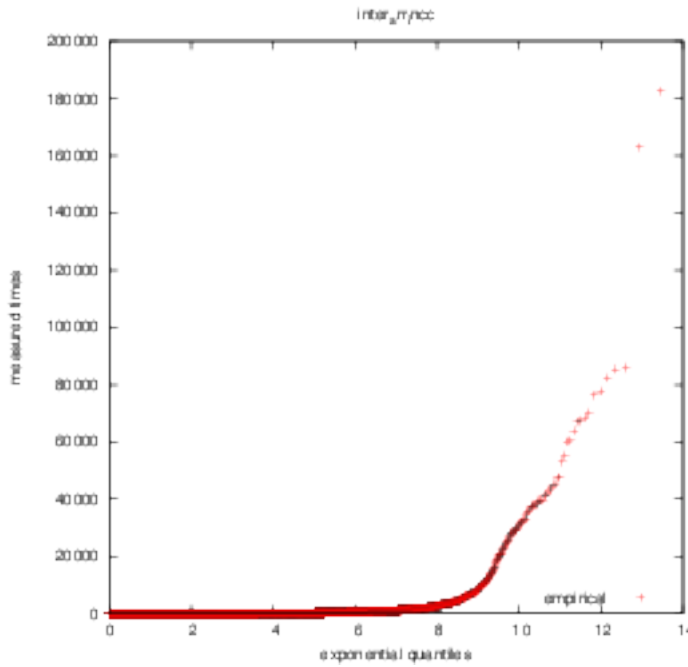

Figure 17. Q-Q plot for INCC jobs' inter-arrival time

In our investigation, we assume that the administration time of jobs takes 10 percent of the execution time of jobs. In Figures 21, 22 and 23, results are plotted without considering the administration time (labeled 'Non-AT') and with administration time (labeled 'AT'). From the results it is observed that the same conclusions can be made. Our algorithms could achieve a trade-off between the performance capacity and the energy efficiency of a computational cluster in the presence of realistic workloads as well. Furthermore, the job administration time has almost negligible impacts.

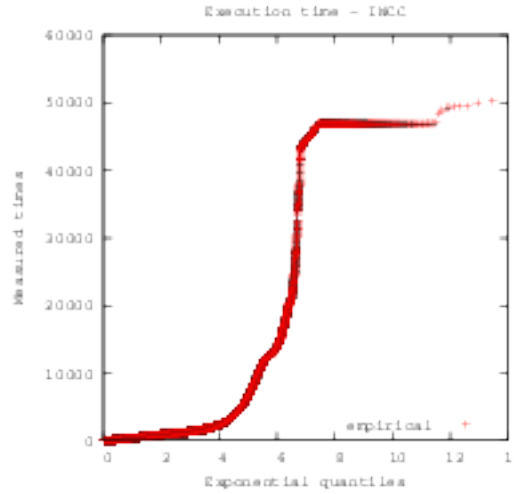

Figure 18. Q-Q plot for INCC jobs' execution time

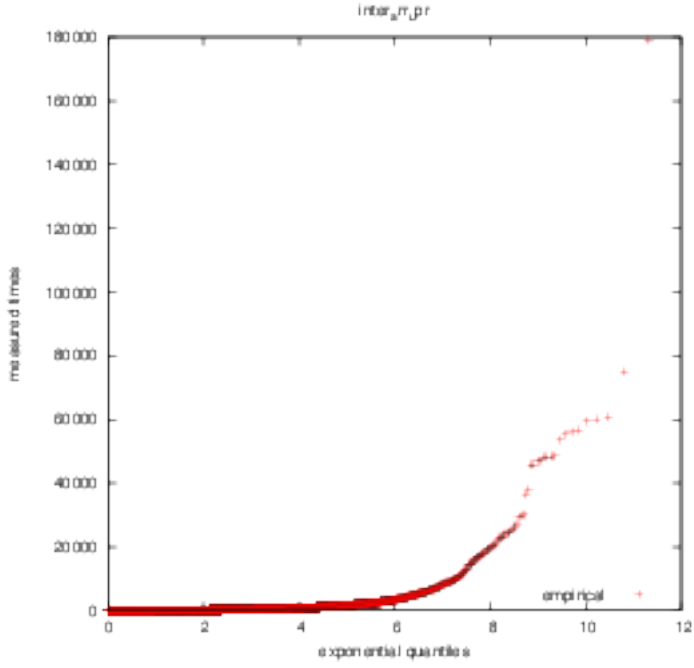

Figure 19. Q-Q plot for UPR jobs' inter-arrival time

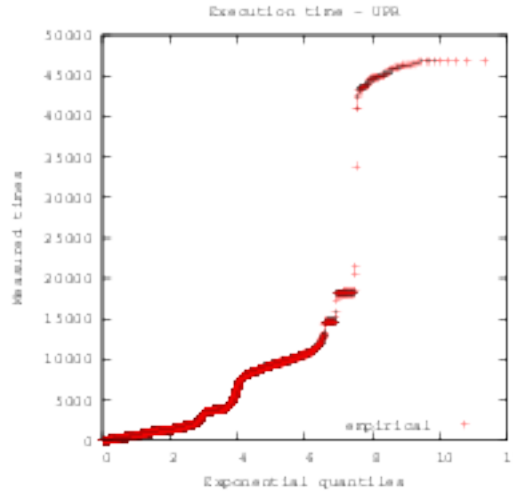

Figure 20. Q-Q plot for UPR jobs' execution time

4 CONCLUSION

In this paper we have proposed two new scheduling algorithms to combine the strengths of the HP policy and the EE policy. The numerical results showed that the new algorithms can be used to find a balance between the performance

Figure 21. Energy consumption (switch-off) with two input data traces

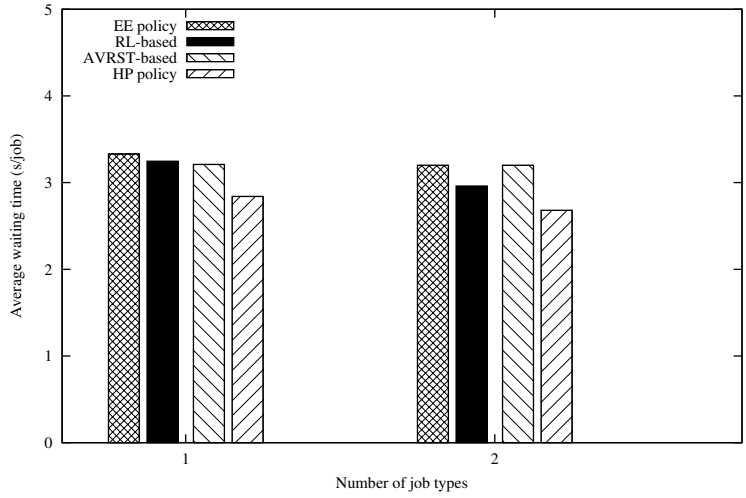

Figure 22. Mean waiting time with two input data traces

capacity and the energy consumption of computational cluster of heterogeneous servers.

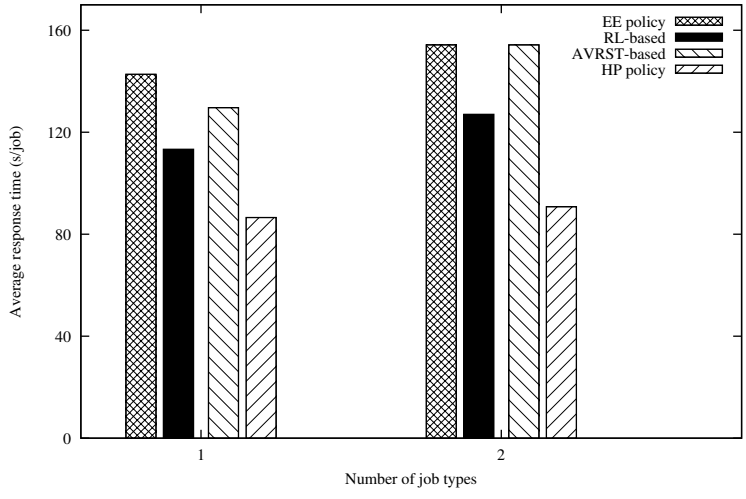

Figure 23. Mean response time with two input data traces

Acknowledgement

The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund. This research is partially supported by the OTKA K123914 project.

REFERENCES

[1] Hluchý, L.: From Grid to Cloud Computing. In: Thang, H. Q., Tran, D. K. (Eds.): SoICT, ACM, 2011, p. 4, doi: 10.1145/2069216.2069220.

[2] Hluchý, L.—Kussul, N.—Shelestov, A.—Skakun, S.—Kravchenko, O. M.—Gripich, Y.—Kopp, P.—Lupian, E.: The Data Fusion Grid Infrastructure: Project Objectives and Achievements. Computing and Informatics, Vol. 29, 2010, No. 2, pp. 319–334.

[3] Nguyen, B. M.—Tran, V. D.—Hluchý, L.: A Generic Development and Deployment Framework for Cloud Computing and Distributed Applications. Computing and Informatics, Vol. 32, 2013, No. 3, pp. 461–485.

[4] Tchernykh, A.—Ramírez, J. M.—Avetisyan, A.—Kuzjurin, N.—Grushin, D.—Zhuk, S.: Two Level Job-Scheduling Strategies for a Computational Grid. Proceedings of the 6th International Conference on Parallel Processing and Applied Mathematics (PPAM '05). Springer-Verlag, Berlin, Heidelberg, LNCS, Vol. 3911, 2006, pp. 774–781, doi: 10.1007/11752578_93.

[5] Yagoubi, B.—Slimani, Y.: Dynamic Load Balancing Strategy for Grid Computing. Transactions on Engineering, Computing and Technology, Vol. 13, 2006, pp. 260–265.

[6] He, Y.—Hsu, W.—Leiserson, C.: Provably Efficient Online Non-Clairvoyant Adaptive Scheduling. Parallel and Distributed Processing Symposium (IPDPS 2007). IEEE International, 2007, pp. 1–10, doi: 10.1109/IPDPS.2007.370303.

[7] Xhafa, F.—Abraham, A.: Computational Models and Heuristic Methods for Grid Scheduling Problems. Future Generation Computer Systems, Vol. 26, 2010, No. 4, pp. 608–621, doi: 10.1016/j.future.2009.11.005.

[8] Zikos, S.—Karatza, H. D.: Resource Allocation Strategies in a 2-Level Hierarchical Grid System. 41st Annual Simulation Symposium (ANSS 2008), 2008, pp. 157–164, doi: 10.1109/anss-41.2008.8.

[9] Zikos, S.—Karatza, H. D.: Communication Cost Effective Scheduling Policies of Nonclairvoyant Jobs with Load Balancing in a Grid. Journal of Systems and Software, Vol. 82, 2009, No. 12, pp. 2103–2116, doi: 10.1016/j.jss.2009.07.006.

[10] Zikos, S.—Karatza, H. D.: The Impact of Service Demand Variability on Resource Allocation Strategies in a Grid System. ACM Transactions on Modeling and Computer Simulation (TOMACS), Vol. 20, 2010, No. 4, pp. 19:1–19:29.

[11] Zikos, S.—Karatza, H. D.: A Clairvoyant Site Allocation Policy Based on Service Demands of Jobs in a Computational Grid. Simulation Modelling Practice and Theory, Vol. 19, 2011, No. 6, pp. 1465–1478.

[12] Zikos, S.—Karatza, H. D.: Performance and Energy Aware Cluster-Level Scheduling of Compute-Intensive Jobs with Unknown Service Times. Simulation Modelling Practice and Theory, Vol. 19, 2011, No. 1, pp. 239–250.

[13] Terzopoulos, G.—Karatza, H. D.: Performance Evaluation of a Real-Time Grid System Using Power-Saving Capable Processors. The Journal of Supercomputing, Vol. 61, 2012, No. 3, pp. 1135–1153, doi: 10.1007/s11227-011-0689-y.

[14] Gkoutioudi, K.—Karatza, H. D.: Multi-Criteria Job Scheduling in Grid Using an Accelerated Genetic Algorithm. Journal of Grid Computing, Vol. 10, 2012, No. 2, pp. 311–323, doi: 10.1007/s10723-012-9210-y.

[15] Tang, X.—Li, K.—Qiu, M.—Sha, E. H.-M.: A Hierarchical Reliability-Driven Scheduling Algorithm in Grid Systems. Journal of Parallel and Distributed Computing, Vol. 72, 2012, No. 4, pp. 525–535.

[16] Do, T. V.—Vu, B. T.—Tran, X. T.—Nguyen, A. P.: A Generalized Model for Investigating Scheduling Schemes in Computational Clusters. Simulation Modelling Practice and Theory, Vol. 37, 2013, pp. 30–42, doi: 10.1016/j.simpat.2013.05.003.

[17] Kumar, U. K.: A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling. International Journal of Computer Science Issues, Vol. 8, 2011, Iss. 5, No. 1, pp. 123–129.

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

[19] SPEC. Acer AW2000h-Aw170h F2 (Intel Xeon e5-2670) Machine. `http://www.spec.org/power_ssj2008/results/res2013q1/power_ssj2008-20121212-00590.html`, February 2013.

[20] SPEC. Acer AW2000h-Aw170h F2 (Intel Xeon e5-2660) machine. `http://www.spec.org/power_ssj2008/results/res2012q4/power_ssj2008-20120918-00546.html`, October 2012.

[21] SPEC. PowerEdge r820 (Intel Xeon e5-4650l) machine. `http://www.spec.org/power_ssj2008/results/res2012q4/power_ssj2008-20121113-00586.html`, November 2012.

Xuan Thi Tran received her B.Sc. degree in electronics and telecommunication engineering from Thai Nguyen University, Vietnam in 2009 and her M.Sc. degree from Budapest University of Technology and Economics. At present she is a Ph.D. student at Network Systems and Services Department, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Hungary. Her research interests are telecommunication networks, cloud computing, performance evaluation and planning of ICT Systems.

Tien Van Do received his M.Sc. and Ph.D. degrees in telecommunications engineering from the Technical University of Budapest, Hungary, in 1991 and 1996, respectively. He is Full Professor in the Department of Networked Systems and Services of the Budapest University of Technology and Economics, a leader of Communications Network Technology and Internetworking Laboratory, and researcher in Division of Knowledge and System Engineering for ICT (KSE-ICT), Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam. He habilitated at BME, and received the D.Sc. from the Hungarian Academy of Sciences in 2011. He has participated and led work packages in the COPERNICUS-ATMIN 1463, the FP4 ACTS AC310 ELISA, FP5 HELINET, FP6 CAPANINA projects funded by EC (where he acted as a work package leader). He led various projects on network planning and software implementations results of which are directly used for industry such as ATM & IP network planning software for Hungarian Telekom, GGSN tester for Nokia, performance testing program for the performance testing of the NOKIA's IMS product, automatic software testing framework for Nokia Siemens Networks. His research interests are queuing theory, telecommunication networks, cloud computing, performance evaluation and planning of ICT Systems.

Binh Thai Vu is a software developer in Analysis, Design and Development of ICT systems (AddICT) Laboratory and Inter-University Cooperative Research Centre, Budapest University of Technology and Economics. His research interest is cloud computing.