Simulation-Based Performance Evaluation of an Energy-Aware Heuristic for the Scheduling of HPC Applications in Large-Scale Distributed Systems

Georgios L. Stavrinides
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
gstavrin@csd.auth.gr

Helen D. Karatza
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
karatza@csd.auth.gr

ABSTRACT

As the distributed resources required for the processing of High Performance Computing (HPC) applications are becoming larger in scale and computational capacity, their energy consumption has become a major concern. Therefore, there is a growing focus from both the academia and the industry on the minimization of the carbon footprint of the computational resources, especially through the efficient scheduling of the workload. In this paper, a technique is proposed for the energy-aware scheduling of bag-of-tasks applications with time constraints in a large-scale heterogeneous distributed system. Its performance is evaluated by simulation and compared with a baseline algorithm. The simulation results show that the proposed heuristic not only reduces the energy consumption of the system, but also improves its performance.

Keywords

Energy-aware scheduling; bag-of-tasks applications; time constraints; large-scale distributed systems; simulation; performance evaluation

1. INTRODUCTION

The inherent and ever-growing need of High Performance Computing (HPC) applications for effective, highly parallel and scalable processing, requires the employment of distributed computing resources that are becoming larger in scale and computational capacity day-by-day. The energy consumption of these platforms has become a major concern. Consequently, there is a growing focus from both the research community and the industry on the minimization of the carbon footprint of the computational resources, especially through the efficient scheduling of the workload.

HPC applications are usually *bags-of-tasks (BoT)*. That is, they consist of independent, embarrassingly parallel tasks that can be executed in any order, without any intertask communication. Examples of such applications include chromosome mapping, pattern matching and data mining applications [14, 15]. Furthermore, such applications often have time constraints (deadlines) within which they must finish execution. A late result in such cases is commonly considered useless [17, 18, 19, 16].

ICPE '17 Companion, April 22-26, 2017, L'Aquila, Italy

DOI: http://dx.doi.org/10.1145/3053600.3053611

1.1 Motivation

The nature and time constraints of the workload, as well as the heterogeneity and energy consumption of the distributed computing resources, constitute major challenges that must be addressed during the scheduling of BoT applications. Effective techniques must be employed, in order to minimize the energy consumption, while maintaining good performance. The performance of such algorithms is usually evaluated via simulation, rather than by analytical methods. Analytical modeling is difficult and often requires simplifying assumptions that may have an unpredictable impact on the results.

1.2 Contribution

In this paper, we propose a technique for the energy-aware scheduling of bag-of-tasks applications with time constraints in a large-scale heterogeneous distributed system. Its performance is evaluated by simulation and compared with a baseline algorithm. The rest of the paper is organized as follows: Section 2 presents an overview of related literature, while Section 3 describes the system, workload and energy consumption models, as well as the proposed scheduling technique. Section 4 presents the performance metrics, the experimental setup and the simulation input parameters. Furthermore, it presents and analyzes the simulation results. Finally, Section 5 concludes the paper, providing directions for further research.

2. RELATED WORK

The scheduling of bag-of-tasks applications has been studied extensively in the literature [9, 5, 23]. Two of the most widely used scheduling heuristics are the *MinMin* and *MaxMin* policies. They were first introduced in [7] and later implemented in a widely distributed environment in [6]. The MinMin strategy is a two-step iterative process. In the first step, the *minimum completion time (MCT)* of each unassigned task is calculated, over all of the processors in the system. In the second step, the task with the minimum MCT is assigned to the corresponding processor. At each

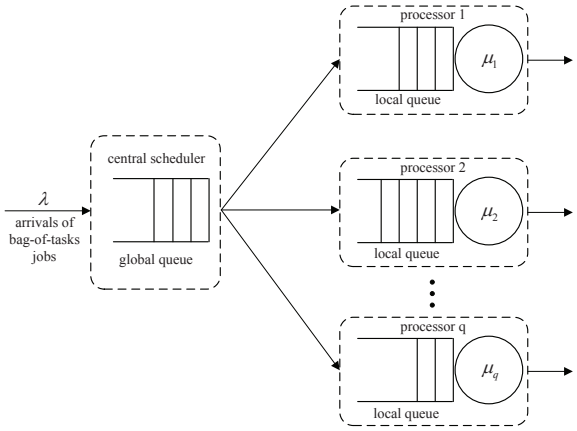

Figure 1: The queueing model of the system under study.

iteration, the current load of the processors is taken into account for the next scheduling decision [25].

The difference between the MaxMin and MinMin policies, is that in the second step of the scheduling process, in the case of MaxMin, the task with the maximum, instead of the minimum, MCT is assigned to the corresponding processor. Consequently, in cases where the application consists of a large number of tasks with small execution times and a few tasks with large execution times (e.g. cases where the computational demands of the tasks are exponentially distributed), the MaxMin heuristic is likely to give a smaller total execution time than the MinMin policy, since it schedules the tasks with larger execution times at earlier iterations [21]. Therefore, there is a higher probability that the tasks with smaller execution times that are subsequently scheduled, will be executed in parallel with the tasks with larger execution times. Alternative versions of the above algorithms have been proposed in the literature, in an attempt to make them energy-aware [12]. However, the vast majority of them trades off performance for energy savings [10]. The goal of our proposed scheduling heuristic is to achieve energy savings, without compromising the system performance, as this is crucial in the case of HPC applications.

3. PROBLEM FORMULATION

3.1 System Model

The distributed system under study consists of a set $Q = \{p_1, p_2, ..., p_q\}$ of q fully connected heterogeneous processors. Each processor p_i serves its own local queue of tasks (i.e. it has its own memory) and has an execution rate μ_i. A dedicated processor acts as a central scheduler, responsible for scheduling the component tasks of the application jobs to the other processors [26]. The queueing model of the target system is shown in Figure 1.

3.2 Workload Model

In our study, the jobs generated by HPC applications arrive at the central scheduler in a Poisson process with rate λ [8, 22, 20]. It is assumed that each job is a bag-of-tasks

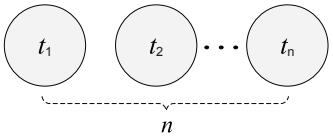

Figure 2: A bag-of-tasks job, consisting of n independent parallel tasks.

(BoT), consisting of a set $J = \{t_1, t_2, ..., t_n\}$ of n independent parallel tasks, without any communication or precedence constraints among them. It is assumed that any input data required by a task to start execution are available on all processors. Furthermore, it is assumed that the component tasks of the jobs are not preemptible, as it has been shown that preemption of time-constrained tasks may eventually lead to performance degradation [4]. Each component task t_i has a weight w_i, which denotes its *computational volume*, i.e. the amount of computational operations that must be executed. The computational volume of the tasks is exponentially distributed with mean $\overline{w}$.

The *computational cost* $Comp$ of a task t_i on a processor p_j is defined as:

$$Comp(t_i, p_j) = w_i/\mu_j \tag{1}$$

where μ_j is the execution rate of processor p_j. The *length* L of a job is equal to the largest average computational cost of its component tasks. That is:

$$L = \max_{t_i \in J} \left\{ \overline{Comp(t_i)} \right\} \tag{2}$$

where $\overline{Comp(t_i)}$ is the average computational cost of task t_i over all of the processors in the system.

It is assumed that each job has an *end-to-end deadline* D, which is given by:

$$D = AT + RD \tag{3}$$

where AT is the arrival time of the job. RD is the *relative deadline* of the job and is uniformly distributed in the range $[2L, 4L]$. When a job fails to complete its execution within its deadline, it is considered lost. An example of a BoT job is illustrated in Figure 2.

3.3 Energy Consumption Model

In a large-scale distributed system, typically the processors consume the greatest amount of energy, compared to the other system resources [24]. Consequently, in this paper we focus on the energy consumption of the processors. Specifically, it is assumed that the power P^i consumed by a processor p_i in the system, is given by:

$$P^i\left(u_i\left(\tau\right)\right) = P^i_{idle} + \left(P^i_{max} - P^i_{idle}\right) \cdot u_i\left(\tau\right) \tag{4}$$

where P^i_{idle} is the power consumed by the processor when idle, whereas P^i_{max} is the power consumed by the processor at 100% utilization. $u_i(\tau)$ is the utilization of the processor and it is a function of time [3, 2, 13]. Therefore, the energy consumption E^i of a processor p_i over a period of time $[\tau_0, \tau_1]$

can be defined as an integral of its power consumption P^i:

$$E^i = \int_{\tau_0}^{\tau_1} P^i\left(u_i\left(\tau\right)\right) d\tau \quad (5)$$

3.4 Scheduling Algorithms

3.4.1 Baseline Policy

As a baseline algorithm, the MaxMin policy is employed, since it provides better results for jobs with tasks that have exponential computational demands, as described in Section 2. We adapted this policy, in order to take into account the deadlines of the jobs during the scheduling process. Specifically, when a job arrives at the central scheduler, it enters a global waiting queue and the scheduler is invoked. In the first step of the first iteration of the scheduling process, the *estimated completion time CT* of each component task t_i of the job is calculated on each processor p_j in the system:

$$CT(t_i, p_j) = \tau_{idle}(t_i, p_j) + Comp(t_i, p_j) \quad (6)$$

where $\tau_{idle}(t_i, p_j)$ is the time at which processor p_j will be able to execute task t_i.

For the calculation of the $\tau_{idle}(t_i, p_j)$ parameter, we first find the potential position of task t_i in p_j's queue, according to the deadline D of t_i's job, so that the task with the earliest deadline is placed at the head of p_j's queue. Consequently, the local waiting queues of the processors are arranged according to the *Earliest Deadline First (EDF)* policy [11]. Subsequently, the *minimum estimated completion time MCT* of each task t_i over all of the processors in the system is calculated, as follows:

$$MCT(t_i) = \min_{p_j \in Q} \{CT(t_i, p_j)\} \quad (7)$$

In the second step of the first iteration of the scheduling process, we select the task with the maximum MCT and we assign it to the processor that can provide it with it.

Subsequently, a second iteration of the scheduling process is performed, taking into account the current load of the processors, as resulted by the scheduling decision of the previous iteration. The iterations continue, until all of the component tasks of the newly-arrived job are scheduled. The pseudocode for the baseline MaxMin policy is shown in Algorithm 1.

3.4.2 Energy-Aware Scheduling Heuristic

Our proposed energy-aware scheduling heuristic is based on the MaxMin policy, which was modified so that the energy consumption of the processors is taken into account during the scheduling process. Specifically, during the first step of the process, along with the estimated completion time CT of each component task t_i on each processor p_j, the *estimated energy consumption* E^j of each processor is also calculated, as follows:

$$E^j = P^j_{max} \cdot (CT(t_i, p_j) - \tau) \quad (8)$$

where P^j_{max} is the power consumption of processor p_j at 100% utilization, whereas τ is the current time. That is, we calculate the estimated amount of energy required to finish the execution of all of p_j's assigned tasks, including task t_i. Subsequently, after calculating the minimum estimated completion time MCT of task t_i as in the case of MaxMin, we find the set of processors that can provide the task with estimated completion time that falls in the range $[MCT\left(t_i\right), MCT\left(t_i\right) \cdot (1 + SM)]$, where SM is the *selection margin* of the algorithm. From this set, we choose the processor p_j that has the minimum estimated energy consumption. The minimum estimated completion time of the task is now $MCT'\left(t_i\right) = CT\left(t_i, p_j\right)$.

The second step of the iteration is performed in the same manner as in the case of MaxMin. That is, we choose the task with the maximum MCT' and we assign it to the processor that can provide it with it. The pseudocode for the proposed ENRG-MaxMin policy is shown in Algorithm 2.

Algorithm 1 The baseline MaxMin policy.

Input: A bag-of-tasks job $J = \{t_1, t_2, ..., t_n\}$ of n independent parallel tasks and a set $Q = \{p_1, p_2, ..., p_q\}$ of q heterogeneous processors.
Output: A schedule of J onto Q.

```
1: repeat
2:    /* 1st Step */
3:    for each task t_i ∈ J do
4:       for each processor p_j ∈ Q do
5:          CT(t_i, p_j) ← τ_idle(t_i, p_j) + Comp(t_i, p_j)
6:       end for
7:       MCT(t_i) ← min_{p_j ∈ Q} {CT(t_i, p_j)}
8:    end for
9:    /* 2nd Step */
10:   select t_i ∈ J so that
      MCT(t_i) = max_{t_m ∈ J} {MCT(t_m)}
11:   assign t_i to p_j where MCT(t_i) = CT(t_i, p_j)
12:   mark t_i as scheduled and remove it from J
13: until all tasks t_i ∈ J are scheduled
```

Algorithm 2 The proposed ENRG-MaxMin policy.

Input: A bag-of-tasks job $J = \{t_1, t_2, ..., t_n\}$ of n independent parallel tasks and a set $Q = \{p_1, p_2, ..., p_q\}$ of q heterogeneous processors.
Output: A schedule of J onto Q.

```
1: repeat
2:    /* 1st Step */
3:    for each task t_i ∈ J do
4:       for each processor p_j ∈ Q do
5:          CT(t_i, p_j) ← τ_idle(t_i, p_j) + Comp(t_i, p_j)
6:          E^j ← P^j_max · (CT(t_i, p_j) − τ)
7:       end for
8:       MCT(t_i) ← min_{p_j ∈ Q} {CT(t_i, p_j)}
9:       for each processor p_j ∈ Q do
10:         if CT(t_i, p_j) ∈ [MCT(t_i), MCT(t_i) · (1 + SM)]
            then
11:            insert p_j ∈ Q'
12:         end if
13:      end for
14:      select p_j ∈ Q' so that E^j = min_{p_n ∈ Q'} {E^n}
15:      MCT'(t_i) ← CT(t_i, p_j)
16:   end for
17:   /* 2nd Step */
18:   select t_i ∈ J so that
      MCT'(t_i) = max_{t_m ∈ J} {MCT'(t_m)}
19:   assign t_i to p_j where MCT'(t_i) = CT(t_i, p_j)
20:   mark t_i as scheduled and remove it from J
21: until all tasks t_i ∈ J are scheduled
```

4. SIMULATION-BASED PERFORMANCE EVALUATION

4.1 Performance Metrics

In order to compare the performance of the proposed energy-aware scheduling heuristic with the baseline algorithm, the following metrics were used:

- *Job Guarantee Ratio*, which is the ratio of the number of jobs that completed their execution within their deadline, over the number of all of the jobs that arrived at the central scheduler, during the observed time period.
- *Total Energy Consumption*, which is the total energy consumed by all of the processors in the system, during the observed time period. It is measured in kilowatt-hours (kWh).

4.2 Simulation Setup

In order to evaluate the performance of the proposed scheduling heuristic in the framework under study, we implemented our own discrete-event simulator in C++, tailored to the specific requirements of the particular problem. Furthermore, no specific workload traces were used, but synthetic workload instead, in order to obtain unbiased, general results, not applicable only to particular workload traces. The simulation input parameters are shown in Table 1. The ENRG-MaxMin strategy was compared to the baseline policy, MaxMin, under different values of the selection margin SM. Specifically, in the case of the ENRG-MaxMin heuristic, the selection margin in the first step of the scheduling process was chosen to be $SM = 5\%, 15\%, 25\%, 35\%$ and 45%.

In this study, it is assumed that half of the processors in the system are small, in terms of computational capacity, with identical execution rate and power consumption characteristics. Correspondingly, the other half of the processors are considered to be large. Specifically, the execution rate and the power consumption characteristics of the small and large processors were modeled after the Intel® Xeon® E3-1260L v5 and Intel® Xeon® E5-4669 v3 processors, respectively, according to the SPECpower_ssj® 2008 benchmark, using the average values per processor for results published in the period 1 Jan. 2015 - 31 Dec. 2016. The SPECpower_ssj® 2008 benchmark is the first industry-standard benchmark for measuring both the performance and the power consumption of servers [1].

The execution rate of a small processor is $\mu_s = 2.6 \cdot 10^5$ SSJ_OPS (Server-Side Java Operations Per Second), whereas the execution rate of a large processor is $\mu_l = 7.9 \cdot 10^5$ SSJ_OPS. The power consumption of a small processor is $P^s_{idle} = 13.8$ W when idle and $P^s_{max} = 46.2$ W at 100% utilization. On the other hand, the power consumption of a large processor is $P^l_{idle} = 23.97$ W and $P^l_{max} = 150.14$ W, respectively. The mean computational volume of the component tasks of the jobs was chosen to be $\overline{w} = 10^8$ SSJ_OPs (Server-Side Java Operations), so that on average, the computational cost of a task on a large processor is approximately equal to 2 minutes. The value of the arrival rate λ of the jobs was chosen to be $\lambda = 0.02$, so that the system is stable.

We ran 30 replications of the simulation with different seeds of random numbers, for each set of input parameters.

Table 2: The improvement in Job Guarantee Ratio and Total Energy Consumption in the case of the proposed ENRG-MaxMin scheduling strategy, over the baseline policy, MaxMin.

Selection margin (%)	Job Guarantee Ratio Increase (%)	Total Energy Consumption Decrease (%)
5	0.0516	0.0527
15	0.1327	0.1501
25	0.1671	0.2084
35	0.1351	0.2023
45	0.1007	0.2052

Each replication was terminated when 10^6 jobs had been completed. For every mean value, a 95% confidence interval was evaluated. The half-widths of all of the confidence intervals were less than 5% of their respective mean values.

4.3 Simulation Results

The simulation results of the comparison of the proposed scheduling heuristic, ENRG-MaxMin, with the baseline policy, MaxMin, are shown in Figures 3 and 4. It can be observed that the proposed technique outperforms the baseline algorithm, in terms of both performance metrics, Job Guarantee Ratio and Total Energy Consumption. Furthermore, it can be observed that the ENRG-MaxMin policy yields the highest guarantee ratio and the lowest energy consumption, when the selection margin SM used in its first scheduling step is set to 25%. Specifically, as it is shown in Table 2, the increase in guarantee ratio is 0.1671%, whereas the decrease in energy consumption is 0.2084%, compared to the baseline strategy. Even though the improvement is relatively small, it can still have a significant impact on the long-term performance and energy consumption of the system.

For higher selection margin values, the performance of the proposed policy starts to deteriorate. This is due to the fact that for higher values, more processors are considered for selection. These processors can provide a higher (i.e. worse) estimated completion time for the task to be scheduled, than the processor that can provide the task with the minimum estimated completion time MCT. From the set of these processors, the algorithm selects the one that requires the minimum estimated amount of energy to finish the execution of all of its assigned tasks, including the task to be scheduled, without taking into account the estimated completion time that it can provide. Consequently, there is a higher probability that a processor with a much higher estimated completion time than the MCT will be selected, ultimately leading to a higher number of jobs missing their deadline. This also results in higher energy consumption, since the energy consumed for the processing of jobs that eventually miss their deadline (i.e. jobs that are eventually lost) is essentially wasted. However, despite the minor degradation in the performance of the ENRG-MaxMin policy for higher selection margin values, it still outperforms the baseline policy, MaxMin.

5. CONCLUSIONS

In this paper, the ENRG-MaxMin heuristic was proposed for the energy-aware scheduling of bag-of-tasks applications with time constraints in a large-scale heterogeneous dis-

Table 1: Simulation model input parameters.

Parameter	Value
Number of completed jobs	10^6
Total number of processors	$q = 128$
Number of small processors	$q_s = 64$
Number of large processors	$q_l = 64$
Small processor execution rate	$\mu_s = 2.6 \cdot 10^5$ SSJ_OPS
Large processor execution rate	$\mu_l = 7.9 \cdot 10^5$ SSJ_OPS
Number of tasks per job	$n \sim U[1, 64]$
Job arrival rate	$\lambda = 0.02$
Job relative deadline	$RD \sim U[2L, 4L]$
Mean task computational volume	$\overline{w} = 10^8$ SSJ_OPs
Small processor power consumption when idle	$P^s_{idle} = 13.8$ W
Small processor power consumption @ 100% utilization	$P^s_{max} = 46.2$ W
Large processor power consumption when idle	$P^l_{idle} = 23.97$ W
Large processor power consumption @ 100% utilization	$P^l_{max} = 150.14$ W
Scheduling method	MaxMin, ENRG-MaxMin
Selection margin for ENRG-MaxMin	$SM = \{5\%, 15\%, 25\%, 35\%, 45\%\}$

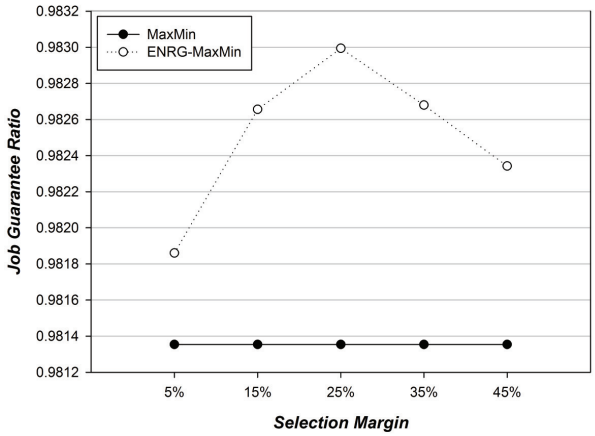

Figure 3: Job Guarantee Ratio in the case of ENRG-MaxMin and MaxMin policies. In the case of the ENRG-MaxMin heuristic, the selection margin in the first step of the scheduling process was chosen to be $SM = 5\%, 15\%, 25\%, 35\%$ and 45%.

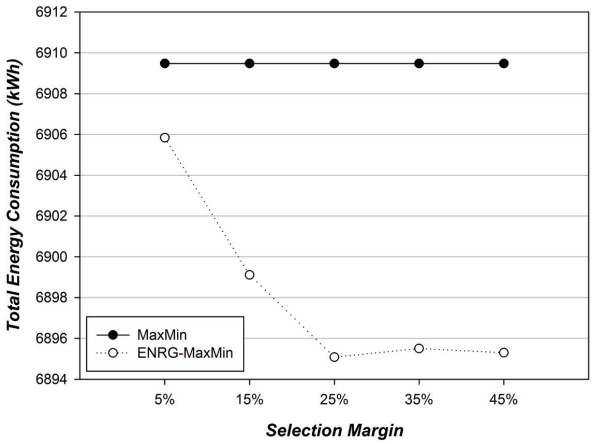

Figure 4: Total Energy Consumption in the case of ENRG-MaxMin and MaxMin policies. In the case of the ENRG-MaxMin heuristic, the selection margin in the first step of the scheduling process was chosen to be $SM = 5\%, 15\%, 25\%, 35\%$ and 45%.

tributed system. Its performance was evaluated by simulation and compared with a baseline algorithm, MaxMin. The simulation results show that the proposed heuristic not only reduces the energy consumption of the system, but also improves its performance. Our future work plans include the evaluation of the proposed heuristic in systems that feature a higher degree of heterogeneity and for workloads with different levels of variability in their computational demands.

6. REFERENCES

[1] Specpower_ssj2008. https://www.spec.org/power_ssj2008/results/power_ssj2008.html. Last update: 21 Dec. 2016.

[2] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, May 2012.

[3] A. Beloglazov and R. Buyya. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science (MGC'10)*, pages 4:1–4:6, Nov. 2010.

[4] G. C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer, 3rd edition, 2011.

[5] R. N. Calheiros and R. Buyya. Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through dvfs. In *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom'14)*, pages 342–349, Dec. 2014.

[6] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel. Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In *Proceedings of the 7th Heterogeneous Computing Workshop (HCW'98)*, pages 184–199, Mar. 1998.

[7] O. H. Ibarra and C. E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM*, 24(2):280–289, Apr. 1977.

[8] H. D. Karatza. Performance of gang scheduling strategies in a parallel system. *Simulation Modelling Practice and Theory*, 17(2):430–441, Feb. 2009.

[9] K. H. Kim, R. Buyya, and J. Kim. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, pages 541–548, May 2007.

[10] Y. Li, Y. Liu, and D. Qian. A heuristic energy-aware scheduling algorithm for heterogeneous clusters. In *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS'09)*, pages 407–413, Dec. 2009.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[12] S. Nesmachnow, B. Dorronsoro, J. E. Pecero, and P. Bouvry. Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of Grid Computing*, 11(4):653–680, Dec. 2013.

[13] N. Quang-Hung, P. D. Nien, N. H. Nam, N. Huynh Tuong, and N. Thoai. A genetic algorithm for power-aware virtual machine allocation in private cloud. In *Proceedings of the 2013 Information and Communication Technology - EurAsia Conference (ICT-EurAsia'13)*, pages 25–29, Mar. 2013.

[14] H. Senger, E. R. Hruschka, F. A. B. Silva, L. M. Sato, C. P. Bianchini, and B. F. Jerosch. Exploiting idle cycles to execute data mining applications on clusters of pcs. *Journal of Systems and Software*, 80(5):778–790, May 2007.

[15] F. A. B. Silva and H. Senger. Scalability limits of bag-of-tasks applications running on hierarchical platforms. *Journal of Parallel and Distributed Computing*, 71(6):788–801, June 2011.

[16] G. L. Stavrinides, F. R. Duro, H. D. Karatza, J. G. Blas, and J. Carretero. Different aspects of workflow scheduling in large-scale distributed systems. *Simulation Modelling Practice and Theory*, 70:120–134, Jan. 2017.

[17] G. L. Stavrinides and H. D. Karatza. The impact of input error on the scheduling of task graphs with imprecise computations in heterogeneous distributed real-time systems. In *Proceedings of the 18th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA'11)*, pages 273–287, June 2011.

[18] G. L. Stavrinides and H. D. Karatza. Scheduling real-time dags in heterogeneous clusters by combining imprecise computations and bin packing techniques for the exploitation of schedule holes. *Future Generation Computer Systems*, 28(7):977–988, July 2012.

[19] G. L. Stavrinides and H. D. Karatza. A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds. In *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud'15)*, pages 231–239, Aug. 2015.

[20] G. L. Stavrinides and H. D. Karatza. Scheduling real-time parallel applications in saas clouds in the presence of transient software failures. In *Proceedings of the 2016 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'16)*, pages 1–8, July 2016.

[21] E. K. Tabak, B. B. Cambazoglu, and C. Aykanat. Improving the performance of independent task assignment heuristics minmin, maxmin and sufferage. *IEEE Transactions on Parallel and Distributed Systems*, 25(5):1244–1256, May 2014.

[22] H. K. Tang, P. Ramanathan, and K. Morrow. Inserting placeholder slack to improve run-time scheduling of non-preemptible real-time tasks in heterogeneous systems. In *Proceedings of the 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems 2014*, pages 168–173, Jan. 2014.

[23] G. Terzopoulos and H. D. Karatza. Power-aware bag-of-tasks scheduling on heterogeneous platforms. *Cluster Computing*, 19(2):615–631, June 2016.

[24] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C. Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry. An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, 16(1):3–15, Mar. 2013.

[25] C. Weng and X. Lu. Heuristic scheduling for bag-of-tasks applications in combination with qos in the computational grid. *Future Generation Computer Systems*, 21(2):271–280, Feb. 2005.

[26] X. Zhu, X. Qin, and M. Qiu. Qos-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters. *IEEE Transactions on Computers*, 60(6):800–812, June 2011.