

Scalable and Energy-efficient Scheduling Techniques for Large-scale Systems

Cesar O. Diaz, Mateusz Guzek, Johnatan E. Pecero, Pascal Bouvry
Computer Science and Communications Research Unit
University of Luxembourg
L-1359 Luxembourg-Kirchberg, Luxembourg
Email: {firstname.lastname}@uni.lu

Samee U. Khan
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND 58108
Email: samee.khan@ndsu.edu

Abstract—The scalability of a computing system can be identified by at least three components: (a) size, (b) geographical distribution, and (c) administrative constraints. Newer paradigms, such as clouds, grids, and clusters bring in more parameters to the aforementioned list, namely heterogeneity, energy consumption, and transparency. To optimize the performance of a computing system, it is manner that exploits heterogeneity and is scalable. Moreover, newer systems also demand energy efficiency as an integral part of schedulers. In this paper, we evaluate the behavior of low complexity energy-efficient algorithms for scheduling. The set of experimental results showed that the evaluated heuristics perform as efficiently as related approaches; demonstrating their applicability and scalability for the considered problem.

Keywords—scalability; heterogenous computing systems; energy conservation; scheduling; performance of systems;

I. INTRODUCTION

Heterogenous computing systems (HCS) are widely used as a not expensive way of obtaining powerful distributed computing systems. One important goal that must met to make building a distributed system worth the effort is scalability [1]. Scalability refers to the capability of the system to adapt to an increase in the service load. The scalability of a system can be measure by at least three different components. First, numerically or with respect of its size. It refers about the feasibility to add more users and resources to the system [1]. The second is geographically scalable system, which is the distance between the farthest nodes within the system. The third feature is the administrative scalability, which refers to manage even if it encompass many independent administrative organizations [2]. In this paper, we consider the first feature.

There have been numerous works in the area of power and energy optimization for such systems [3], [4], [5], [6], [7], [8], [9], [10], [11]. Another dimension of this work is energy conservation, there has been a lot of work of a reason on the afore mentioned topic, which are covered by surveys such as [8], [9], [20], [21].

In this paper, we evaluate the energy-aware scheduling heuristics proposed in [12] by one scalability dimension, its size. This target is achieved by evaluating these scheduling algorithms increasing the number of tasks and machines. The main idea of these heuristics is to match each task with the

best resource to execute the task, that is, the resource that optimizes the completion time of the task and executes the task fastest with minimum energy. These algorithms take advantage of the resource capabilities (i.e. heterogeneity) and task requirements featuring a very low overhead. They were evaluated by adding numerous simulations featuring high heterogeneity of resources, and/or high heterogeneity of applications. Simulations studies are performed to compare these heuristics with the well-known *min-min* [14], [8]. We used *min-min* as a basis of comparison because it is one of the most practical and applicable heuristic in the context of HCS [14], [15], [16], [17]. *Min-min* starts by considering that all tasks are not mapped. It works in two phases. In the first phase, the algorithm establishes the minimum completion time for every unscheduled job. In the second phase, the task with the overall minimum expected completion time is selected and assigned to the corresponding machine. The task is then removed from the set and the process is repeated until all tasks are mapped. The run time of *min-min* is $O(t^2m)$ [19].

We have considered the minimization of the *makespan* (i.e., the maximum completion time) and *energy* as a basis of comparison. We also compare these heuristics based on the *flowtime*. The flowtime of a task is the length of the time interval between the release time and completion time of the task. It is commonly used as a quality of service (QoS) measure that allows to guarantee good response times [18].

This paper is organized as follows. The system, scheduling and energy models are presented in Section II. We briefly describe the evaluated heuristics in Section III. Experimental results are given in Section IV. Section V concludes the paper.

II. MODELS

A. System, Application and Scheduling Models

We consider a heterogeneous computing system composed of a set of $M = \{m_1, \dots, m_m\}$ machines. We consider that resources in the target system are incorporated with an effective energy-saving mechanism for idle time slots [22], [10]. The energy consumption of an idle resource at any given time is set using a minimum voltage based on the processor's architecture. In this paper, we used two different

voltage levels: *maximum*, when the processor is performing work or it is in an active state and *idle* level, when processor is in an idle state. We consider a set of independent tasks $T = \{t_1, \dots, t_n\}$ to be executed onto the system. The tasks are considered as an indivisible unit of workload. Each task has to be processed completely on a single machine. The computational model we consider in this work is the ETC model [23]. The *ETC* matrix of size $t \times m$ it is assumed to be known. Each position $ETC[t_i][m_j]$ in the matrix indicates the expected time to compute task t_i on machine m_j . This model allows to represent the heterogeneity among tasks and machines. For more details we refer the reader to [23], [13] and [12].

The considered scheduling problem is formulated as follows. Given the heterogenous computing systems composed of the set of m machines, and the set of n tasks. Any task is scheduled without preemption from time $\sigma(t_i)$ on machine $\pi(t_i)$, with an execution time $ETC[t_i][m_j]$. The task t_i completes at time C_i equals to $\sigma(t_i) + ETC[t_i][m_j]$. The objective is to minimize the maximum completion time ($C_{max} = \max(C_i)$) or makespan and the total energy E_t used to execute the tasks. Additionally, in this paper we also aim to guarantee good response times. In this context, response time is modeled as flowtime. As we already mentioned, the flowtime of a task is the length of the time interval between the completion time and release time. We consider that the release time is zero for all the tasks. Hence, the flowtime represents the sum of completion time of jobs, that is, $\sum_{i=1}^n C_i$, the aim is to minimize $\sum_{i=1}^n C_i$.

B. Energy Model

The energy model used in this work is derived from the power consumption model in digital complementary metal-oxide semiconductor (CMOS) logic circuitry. The power consumption of a CMOS-based microprocessor is defined to be the summation of capacitive power, which is dissipated whenever active computations are carried out, short-circuit and leakage power (static power dissipation). The capacitive power (P_c) (dynamic power dissipation) is the most significant factor of the power consumption. It is directly related to frequency and supply voltage, and it is defined as [24]:

$$P_c = AC_{eff}V^2f, \quad (1)$$

where A is the number of switches per clock cycle, C_{eff} denotes the effective charged capacitance, V is the supply voltage, and f denotes the operational frequency. The energy consumption of any machines in this paper is defined as:

$$E_c = \sum_{i=1}^n AC_{eff}V_i^2fETC[i][M[i]], \quad (2)$$

where $M[i]$ represents a vector containing the machine m_j where task t_i is allocated, V_i is the supply voltage of the

machine m_j . On the other hand, the energy consumption during idle time is defined as:

$$E_i = \sum_{j=1}^m \sum_{idle_{jk} \in IDLES_j} AC_{eff}V_{min_j}^2I_{jk}, \quad (3)$$

where $IDLES_j$ is the set of idling slots on machine m_j , V_{min_j} is the lowest supply voltage on m_j , and I_{jk} is the amount of idling time for $idle_{jk}$. Therefore, the total energy consumption is defined as:

$$E_t = E_c + E_i. \quad (4)$$

III. LOW COST ALGORITHMS

In the scheduling problem on HCS, near-optimal solutions would suffice rather than searching for optimality for most practical applications. Therefore, we used the three low-cost scheduling algorithms with good quality schedules and low energy consumption developed in [12]. The algorithms are based on the list scheduling approaches and they are considered as batch mode dynamic scheduling heuristics. The main difference between the algorithms is the priority used to construct the list. For that, minimum, maximum and average completion time of the task are used as if it was the only task to be scheduled on the computing system. The name of the heuristics is *MinMin Min* (minimum completion time of tasks sorted in decreasing order of its minimum completion time and scheduled based on the minimum completion time), *MinMax Min* (maximum completion time of tasks sorted in decreasing order of its maximum completion time and scheduled based on the minimum completion time), *MinMean Min* (average completion time of tasks, sorted in decreasing order of the minimum average completion time, and scheduled based on the minimum completion time).

The principle of the algorithms is to assign the tasks not only to the machine which minimizes its expected completion time, but its execution time as well. The authors proposed a weighted function called the *score function* $SF(t_i, m_j)$ (see Eq. 5), that balances both objectives. The rational is to minimize the workload of machines and intrinsically minimize the energy used to carry out the work. The computational complexity of the heuristics is $O(tm \log t)$, which is less than one order of magnitude to the comparative approaches.

The score of each mapping event is calculated as in Eq. 5. For each machine m_j ,

$$SF(t_i) = \lambda \cdot \frac{C_i}{\sum_{k=1}^m C_{ik}} + (1-\lambda) \cdot \frac{ETC[t_i][m_j]}{\sum_{k=1}^m ETC[t_i][m_k]}, \quad (5)$$

where $\sum_{k=1}^m C_{ik}$ is the sum of the completion time of the task t_i over all machines and $\sum_{k=1}^m ETC[t_i][m_k]$ is the sum of the expected time to complete of task t_i over all machines. The first term of equation 5 aims to minimize the completion time of the tasks t_i , while the second term aims to assign the task to the fastest machine or the machine on which the task takes the minimum expected time to complete.

IV. EXPERIMENTAL EVALUATION

We compare the algorithms proposed in [12] and the min-min algorithm by simulation using randomly built ETCs. The ETC model can be characterized by three parameters [13], [23]: (a) the first parameter is machine heterogeneity, on which we can distinguish among *low* and *high* machine heterogeneities, (b) the second parameter is task heterogeneity, we can also distinguish among *low* and *high* task heterogeneities, and (c) the third parameter is the consistency, which tries to reflect the characteristics of realistic scenarios. The three different scenarios are: *consistent*, *inconsistent* and *semi-consistent*. Table I, shows the twelve combinations of heterogeneity types (task and machines) and consistency classifications in the ETC model that we use in this paper. The consistency categories are named for the correspondent initial letter (*c* stands for consistent, *i* for inconsistent, *s* for semi-consistent, *lo* stands for low heterogeneity and *hi* for high heterogeneity). Hence, a matrix named *c_lolo* corresponds to a consistent scenario with low task heterogeneity and low machine heterogeneity.

Table I: ETC consistency models.

Consistency		
Consistent	Semi-consistent	Inconsistent
c_lolo	s_lolo	i_lolo
c_lohi	s_lohi	i_lohi
c_hilo	s_hilo	i_hilo
c_hihi	s_hihi	i_hihi

A. Experiments

For the generation of these ETC matrices we have used the coefficient of variation based method (COV) introduced in [23]. The COV-based method provides a greater control over the spread of the execution time values than the common-based method. To simulate different heterogeneous computing environments we have changed the parameters μ_{task} , V_{task} and $V_{machine}$, which represent the mean task execution time, the task heterogeneity, and the machine heterogeneity, respectively. We have used the following parameters: V_{task} and $V_{machine}$ equal to 0.1 for low case respectively and 0.6 for high case, and $\mu_{task} = 100$. The heterogeneous ranges were chosen to reflect the fact that in real situations there is more variability across the execution time for different tasks on a given machine than that across the execution time for a single task on different machines.

We assume that all of the tasks arrive at the system before the scheduling event. Furthermore, we consider that all the machines are idle or available at time zero, this can be possible by considering advance reservation. The algorithms were evaluated through a large set of instances. The instances were generated with different task size and machines combinations for: 512, 1024, 2048, 4096 and 8192 tasks in size to be scheduled each one on 16, 32, 64, 128

and 256 machines. We have generated 30000 instances, 100 instances for each 12 cases and for each 25 task and machine combinations. Additionally, we have considered different voltages for the machines. We randomly assigned these voltages to machines by choosing among three different set. The first set considers 1.95 and 0.8 Volts for maximum or active state and minimum or idle state, respectively. The second set is 1.75 Volts at maximum state and 0.9 Volts at idle state. Finally, the last set considers 1.6 Volts at maximum level and 0.7 Volts at idle level.

B. Results and Discussions

The results for the algorithms are depicted from Figs. 1 to 4. We show normalized values of makespan, flowtime and energy for each heuristic against min-min for λ -values in the interval [0, 1]. The normalized data were generated by dividing the results for each heuristic by the maximum result computed by these heuristics. We only show the plots for the high task and high machine heterogeneity for the three different scenarios, in reference to [12] and to investigate the scalability we show the most significant results. The legends *m-m n_mksp*, *m-m n_flow* and *m-m n_rcb* in the figures stand for makespan, flowtime and energy of min-min respectively.

We can observe that the behavior for the three evaluated algorithms were quite similar, this can be evidenced in Fig. 1 for the 1024x32 s_hihi. This results show the similar behavior of the evaluated algorithms [12] for this reason, we only show, in all the next figures, the results for the minMin_min algorithm against min-min. As can be seen in Fig. 1 and it can be verified in Table IV, the value of $\lambda=0.4$ is the best value for the three considered objectives, makespan, flowtime and energy. We obtained all the representative values of Tables II, III and IV following the same method. These tables show that with one value of λ the evaluated heuristics can perform as well as min-min for all the three considered metrics, even, sometimes better. These λ -values can be verified from Figs. 2 through 4.

Table II: Lambda Values for c_hihi

Machines	Tasks				
	512	1024	2048	4096	8192
16	0.8	0.7	0.7	0.7	0.7
32	0.6	0.6	0.6	0.6	0.7
64	0.5	0.6	0.6	0.5	0.6
128	0.4	0.5	0.5	0.4	0.5
256	0.2	0.3	0.4	0.4	0.3

It can be observed from these figures that the evaluated heuristics follow the same performance behavior according to the different scenarios. Relative values range are biggest for the consistent instances than semi-consistent and inconsistent. The results clearly demonstrate that energy efficiency is the best for the consistent instances in most of the experiments. It may be related to the fact, that the makespan has worse results. However, for values of λ than

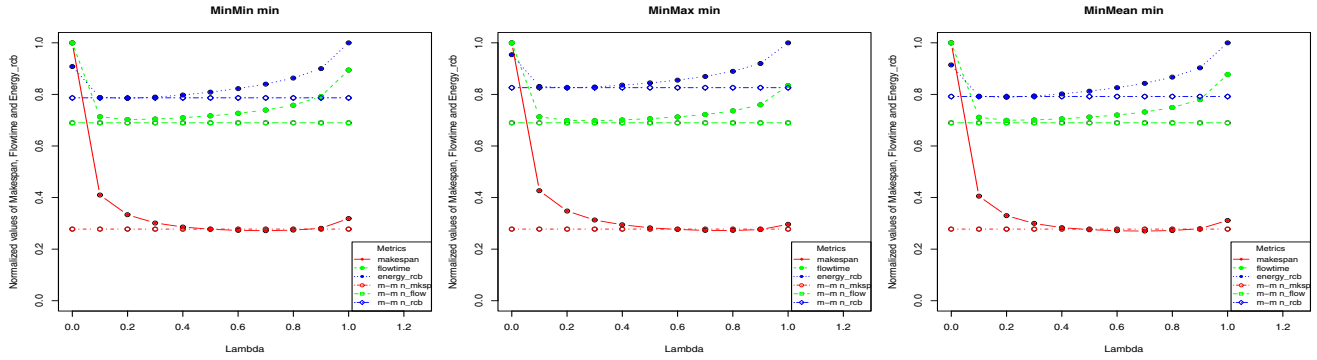


Figure 1: Relative performances of the schedules produced by different heuristics 1024x32 in the s_hihi instances.

Table III: Lambda Values for i_hihi

<i>Machines</i>	<i>Tasks</i>				
	512	1024	2048	4096	8192
16	0.2	0.3	0.3	0.3	0.3
32	0.2	0.2	0.2	0.2	0.3
64	0.2	0.2	0.2	0.2	0.2
128	0.2	0.2	0.2	0.2	0.2
256	0.3	0.2	0.2	0.2	0.1

Table IV: Lambda Values for s_hihi

<i>Machines</i>	<i>Tasks</i>				
	512	1024	2048	4096	8192
16	0.5	0.5	0.5	0.6	0.7
32	0.4	0.4	0.4	0.5	0.6
64	0.3	0.3	0.3	0.4	0.5
128	0.3	0.3	0.3	0.3	0.4
256	0.4	0.3	0.2	0.2	0.3

we showed in Tables II through IV and it can be validated in Figures 2 through 4, we can also observe that the evaluated algorithms can improve makespan and flowtime results by λ in all instances. Interestingly, if the instance is more inconsistent, the evaluated algorithms perform better.

Observing Tables II through IV, we can see the best values for λ , in the c_hihi instances. This value is not changing at all when the number of machines is fixed and the number of tasks increases, as it is displayed in Fig. 3, by scalability measures, one of the size components increase. By other hand, when the number of machines increases and the tasks is fixed, the λ value changes because the makespan and the flowtime improve with low values and the energy for these values always are better (i.e. in all the experiments of this paper, without taking account the other two metrics, energy has the best values in $\lambda=[0.1, 0.3]$) as can be observed in Fig. 2.

The behavior of evaluated algorithms improves, when the number of machines increases considering the inconsistent scenario with high task and machine heterogeneity instances. These results can be observed in from plots in Fig. 4. the

best values of λ are showed in Table III. The benefit of exploiting the heterogeneity of the applications and resources to maximize the performance of the system and energy is more apparent. This is mainly because these instances are the ones presenting the highest inconsistency and heterogeneity.

For the semi-consistent scenario the behavior of the algorithms is like between consistent and inconsistent scenarios as it can be establish in Table IV. In terms of flowtime, all the heuristics are as efficient as min-min, however, the evaluated heuristics have lower complexity.

C. Extended Experimented Evaluations

In the previous experiments (as described in Section IV.B) we have evaluated the energy consumption of the provided solutions by the different algorithms. We are also interested in the energy consumed by the scheduling algorithms itself to provide the schedules. For that, as a preliminary experiment we have measured the energy consumed when executing the investigated algorithms on different instances' sizes. We used "Dominion PX" (outlet metered, outlet switched iPDU - PX-500) device for measure the power consumption at any time to the one machine running the evaluated algorithm [25]. Dominion PX is a power distribution unit (PDU) that offers real-time remote unit-level and individual outlet-level switching and power monitoring of current (amps), voltage, power (kVA, kW), power factor and energy consumption (kWh) with ISO/IEC +/- 1% billing-grade accuracy. The experiments were performed in a machine Intel(R) Pentium 4 CPU 1500MHz, 512 MB in RAM, core voltage of 1.75V. We have only considered the following instances: 512x16, 1024x32, 2048x64, 4096x128, 8192x256. We have executed the algorithms over these sets of instances and we have considered the average consumed energy for each size instance. As mentioned previously, each set is composed of 100 instances. Each algorithm has been executed separately and the experiments are independent. We show the preliminary results of the experiment in Fig. 5. We have used logarithmic scale to emphasize the results. As can be observed, the fast algorithms have the same

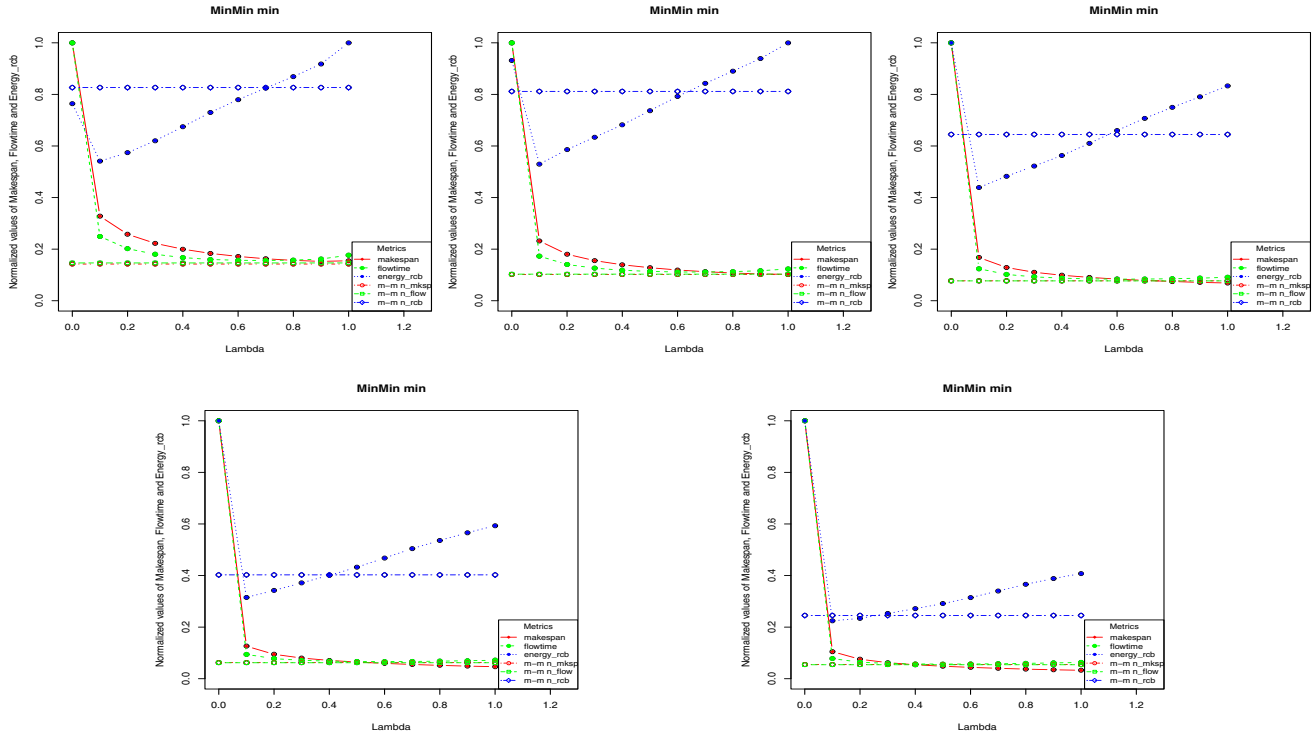


Figure 2: Relative performances of the schedules produced by different heuristics in the *c_hihi* instances with 512 tasks and from top left to bottom right 16, 32, 64, 128 and 256 machines, respectively.

behavior, that is, these algorithms use almost the same energy to compute a schedule over all the set of instances. These heuristics outperform min-min for all the sizes of the instances and the gain in energy-efficiency is more important when the size of the instances scales. The complexity of min-min is one of the main factors that makes the algorithm more energy-inefficient regarding the fast heuristics. Moreover, min-min uses much more memory when the instances scale because the algorithm always evaluate the completion time for the remaining tasks to be scheduled at each step of the loop, because of that the energy consumption increases. On the contrary, the fast heuristics have a good scalability and low overhead than min-min. The usage memory is reduced, because at each step only consider one task to be scheduled, the task with the highest priority, hence their are more energy-efficient than min-min.

V. CONCLUSIONS

This paper investigated three batch mode scheduling algorithms in the context of performance, energy efficiency and scalability in HCS. The set of experimental results showed that the investigated heuristics perform as efficiently as the related approach although featuring lower complexity, lower running time, showing their applicability for the considered scheduling problem and their good scalability.

As part of future work, we intend to implement the

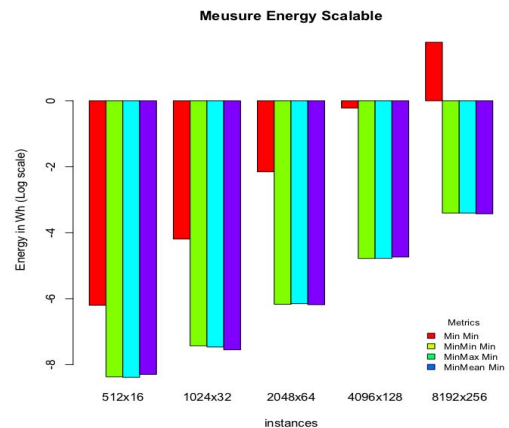


Figure 5: Measure of the energy consumed by the different algorithms.

evaluated heuristics in a packet-level simulator of energy-aware cloud computing data center, GreenCloud [6] and we plan to extend these heuristics to include thermal aspects. Additionally, we consider using dynamic frequency and voltage scaling techniques.

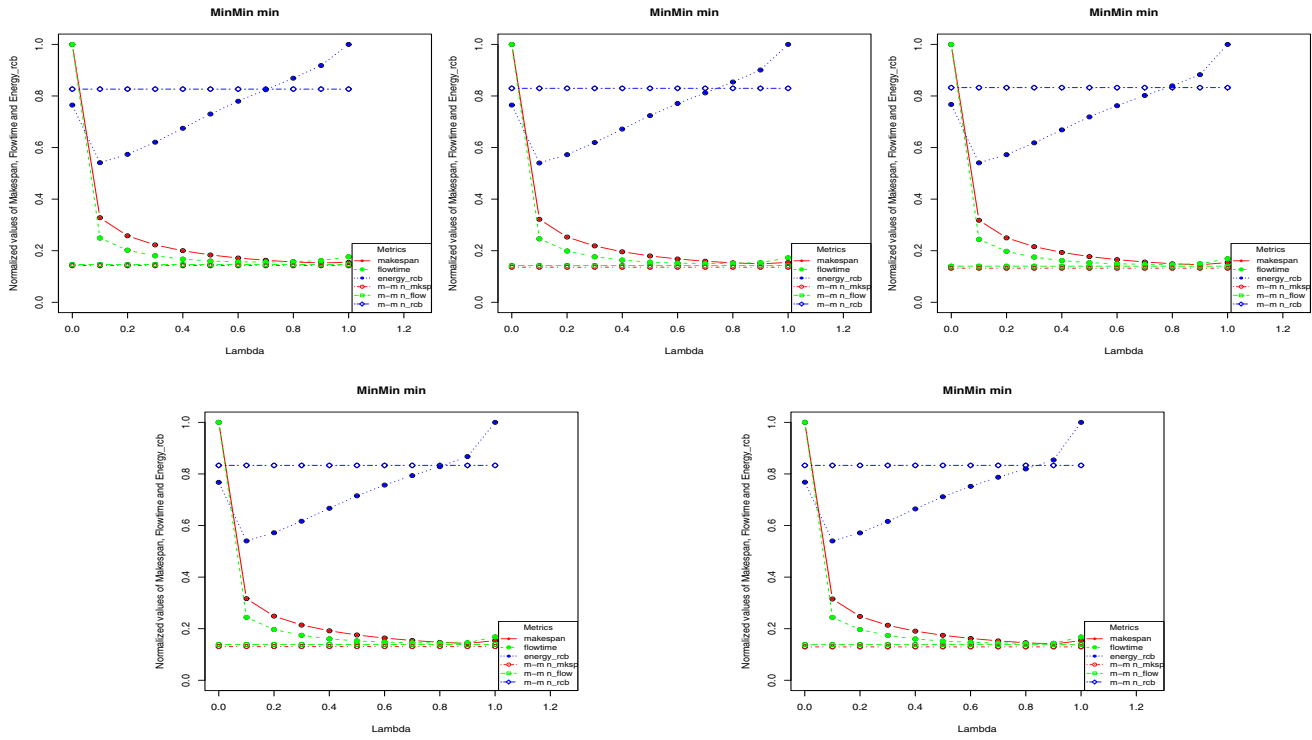


Figure 3: Relative performances of the schedules produced by different heuristics in the *c_hihi* instances with 16 machines and from top left to bottom right 512, 1024, 2048, 4096 and 8192 tasks, respectively.

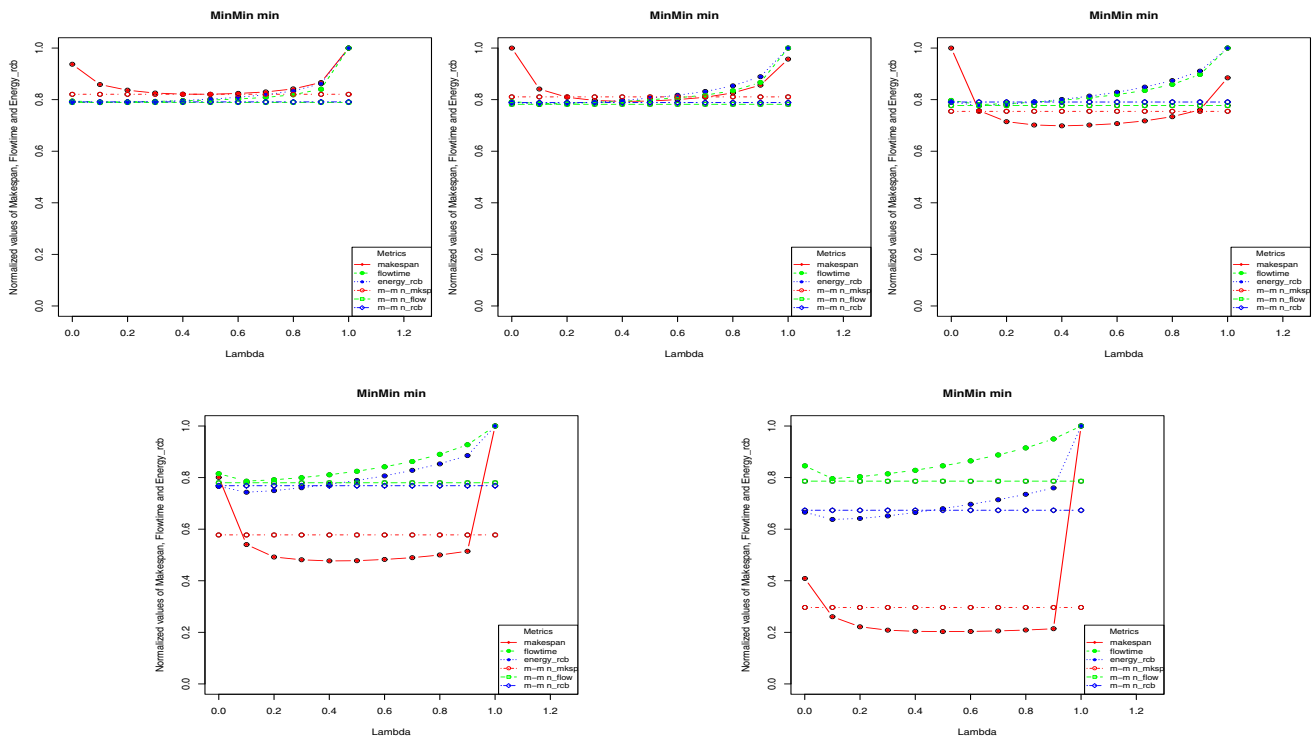


Figure 4: Relative performances of the schedules produced by the different heuristics in the *i_hihi*.

ACKNOWLEDGMENT

This work is supported by the National Research Fund (FNR) of Luxembourg through project Green-IT no. C09/IS/05 and the Luxembourg research grant scheme (AFR) through the grant no. PDR-08-010.

REFERENCES

- [1] A. S. Tanenbaum and M. V. Steen, *Distributed Systems: Principles and Paradigms*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.
- [2] B. C. Neuman, "Scale in distributed systems," in *Readings in Distributed Computing Systems*. Los Alamitos, CA, USA, 1994, pp. 463–489.
- [3] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Symposium on Operating Systems Principles*, vol. 35, Canada, 2001, pp. 103–116.
- [4] P. Lindberg, J. Leingang, D. Lysaker, S. U. Khan, and J. Li, "Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems," *The Journal of Supercomputing*, vol. 53, 2010.
- [5] S. Khan and I. Ahmad, "A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids," *IEEE Trans on Parallel and Dist Systems*, vol. 20, no. 3, pp. 346–360, 2009.
- [6] D. Kliazovich, P. Bouvry, and S. U. Khan, "Dens: Data center energy-efficient network-aware scheduling," in *The 2010 IEEE/ACM Int Conf on Green Computing and Communications (Greencom)*, China, 2010.
- [7] T. Heath, B. Diniz, E. V. Carrera, W. J. Meira, and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proc of the tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPOPP '05. New York, NY, USA: ACM, 2005, pp. 186–195.
- [8] J.-K. Kim, H. J. Siegel, A. A. Maciejewski, and R. Eigenmann, "Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling," *IEEE Trans. on Parallel Distr. Syst.*, vol. 19, pp. 1445–1457, 2008.
- [9] Y. Li, Y. Liu, and D. Qian, "A heuristic energy-aware scheduling algorithm for heterogeneous clusters," in *ICPADS*. IEEE, 2009, pp. 407–413.
- [10] Y. Lee and A. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, pp. 1–13, 2010.
- [11] M. Guzek, J. E. Pecero, B. Dorrnsoro, P. Bouvry, and S. U. Khan, "A cellular genetic algorithm for scheduling applications and energy-aware communication optimization," in *HPCS*, 2010, pp. 241–248.
- [12] C. O. Diaz, M. Guzek, J. E. Pecero, G. Danoy, P. Bouvry, and S. U. Khan, "Energy-aware fast scheduling heuristics in heterogeneous computing systems," in *Procs of the ACM/IEEE/IFIP Int Conf on High Performance Computing and Simulation*, ser. HPCS 2011, Turkey, 2011.
- [13] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, pp. 810–837, 2001.
- [14] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *J. ACM*, vol. 24, pp. 280–289, 1977.
- [15] 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 *Procs of the Seventh Heterogeneous Computing Workshop*, ser. HCW '98. USA, 1998.
- [16] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli, "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," *J. Parallel Distrib. Comput.*, vol. 67, pp. 154–169, 2007.
- [17] F. Pinel, J. E. Pecero, P. Bouvry, and S. U. Khan, "A two-phase heuristic for the scheduling of independent tasks on computational grids," in *Procs of the ACM/IEEE/IFIP Int Conf on High Performance Computing and Simulation*, ser. HPCS 2011, Turkey, 2011.
- [18] S. Albers, "Energy-efficient algorithms," *Commun. ACM*, vol. 53, pp. 86–96, 2010.
- [19] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proc of the Eighth Heterogeneous Computing Workshop*, ser. HCW '99. USA, 1999.
- [20] S. Zeadally, S. U. Khan., and N. Chilamkurti, "Energy-efficient networking: Past, present, and future," *Journal of Supercomputing*, Forthcoming.
- [21] M. A. Aziz., S. U. Khan., T. Loukopoulos., P. Bouvry., H. Li., and J. Li, "An overview of achieving energy efficiency in on-chip networks," *Intl Journal of Communication Networks and Distributed Systems*, vol. 5, pp. 444–458, 2010.
- [22] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: Eliminating server idle power," *SIGPLAN Not.*, vol. 44, pp. 205–216, 2009.
- [23] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Representing task and machine heterogeneities for heterogeneous computing systems," *Journal of Science and Engineering*, vol. 3, no. 3, pp. 195–207, 2000.
- [24] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *9th IEEE/ACM Int Symposium on Cluster Computing and the Grid*, 2009, pp. 92–99.
- [25] "Outlet metered, outlet switched ipdu – px-5000." Available: <http://www.raritan.com/products/power-management/px-5000/>