On The Energy Optimization for Precedence Constrained Applications Using Local Search Algorithms

Johnatan E. Pecero*, Héctor Joaquín Fraire Huacuja†, Pascal Bouvry*, Aurelio Alejandro Santiago Pineda†, Mario César López Locés†, Juan Javier González Barbosa†
*Computer Science and Communication Research Unit, University of Luxembourg,
L-1359 Luxembourg-Kirchberg, Luxembourg
Email: {firstname.lastname}@uni.lu
†Instituto Tecnológico de Ciudad Madero (ITCM),
Av. 1o. de Mayo s/No esq. Sor Juana Inés de la Cruz, C.P. 89440, Cd. Madero, Tam. México
Email: automatas2002@yahoo.com.mx, alx.santiago@gmail.com, mario.cesar@me.com, jjgonzalezbarbosa@hotmail.com

***Abstract*—We investigate the problem of scheduling precedence constrained applications on a distributed heterogeneous computing system with the aim of minimizing schedule length and reducing energy consumption. We present a scheduling algorithm based on the best-effort idea that promotes local search algorithms and dynamic voltage scaling to reduce energy consumption. The final goal is to maintain a given performance while minimizing energy use. The proposed approach first uses a list-based scheduling algorithm to find near-optimal solutions for schedule length, then local search algorithms with dynamic voltage scaling are applied to reduce energy consumption. However the algorithm it's not allowed to deteriorate the schedule length computed by the best-effort algorithm. We discuss simulation results obtained with sets of real-world applications that emphasize the interest of the approach.**

***Keywords*—Energy conservation; scheduling; performance of systems; optimization; greenIT; heterogenous distributed systems;**

I. INTRODUCTION

Distributed computing systems composed of heterogeneous resources are promising for fast processing of computationally intensive applications with different computation needs. However, energy consumption on these systems is becoming the main operational expense and it is still growing. A recent study by DatacenterDynamics Company [1] estimates that the world's data centers will consume 19% more energy in 2012 than they have in 2011. Heat dissipation has a severe impact on the overall system worsening system reliability, eventually resulting in hardware failure caused by excessive heat, hence in poor system performance. Therefore, the reduction of energy consumption is a basic consideration in the design of modern computing systems.

In this paper, the goal is to minimize energy consumption when scheduling precedence constrained applications on a heterogeneous computing distributed system, while a given scheduling performance is preserved. We consider the minimization of schedule length or makespan as a performance criterion. This problem with minimum makespan and energy is a trade-off between schedule length and energy consumption. The reduction in energy consumption is often made by lowering supply voltage and this results in increasing the task execution time, hence makespan. This is the principle of the Dynamic Voltage Scaling (DVS) technique. DVS (equivalently, dynamic voltage frequency scaling, dynamic speed scaling, dynamic frequency scaling) [2] enables processors to dynamically adjust voltage supply levels aiming to reduce power consumption at the expense of performance deterioration. For heterogeneous distributed computing systems, variable voltage scheduling is more challenging because the supply voltage of the executed tasks must be optimized to maximize energy savings. This requires intelligent slack allocation between tasks. The problem is further complicated because slack allocation must consider variations in workload and energy consumption among different tasks. Moreover, the discrete voltage scaling problem is strongly NP-hard [3], the proof is a reduction to the discrete time-cost trade-off problem by restricting the dynamic voltage scaling problem to contain only tasks that requires an execution of one clock cycle.

Variable voltage scheduling problem is commonly addressed in two ways: by a simultaneous or independent approach. In the simultaneous approach, the scheduling is computed in a global cost function including performance and energy saving to satisfy both makespan and energy constraints at the same time. Different solutions produce trade-offs between the two objectives, which means there is no single optimum solution. In this case, the problem is modeled as a multi-objective multi-constrained optimization problem and the objective becomes finding Pareto optimal schedules (i.e., non-dominated schedules), such that no schedule can be better and use less energy.

In the independent mode, which is considered as a two-phases optimization technique, a best-effort scheduling algorithm is combined with a slack reclamation technique [4]–[6]. The slack reclamation algorithm is used as a second

978-1-4673-2362-8/12/$31.00 ©2012 IEEE

pass to minimize the energy consumption of tasks in a schedule generated by the best-effort scheduling algorithm. The principle of slack reclamation is simply to exploit the slack times during software execution to accommodate the negative impacts on perfomance when applying DVS. The challenge is to make appropriate decisions on processor speeds to guarantee the timing requirements while also considering the timing requirements of all tasks in a system. In this mode, energy and time constraints are independent and new scheduling algorithms or exiting algorithms can be adapted to become energy efficient. This approach corresponds to the lexicographic approach to solve multi-criteria problems.

In this work, we propose a scheduling algorithm based on the best-effort idea that optimizes both makespan and energy objectives. We consider the independent mode approach. That is, the algorithm firstly looks for near-optimal solutions using a list-based scheduling algorithm to find the minimum makespan (best-effort). Then, we fix the makespan as a constraint and we optimize energy. For that, we introduce two new local search algorithms that promote DVS without allowing deterioration on the fixed objective. However, makespan is optimized by allocating a given task to different allocation (i.e., processor). These algorithms are based on the best improvement iterative local search process, however the main difference between them is the way the neighborhood of a given solution is generated changing the search space.

This paper is organized as follows. The system, scheduling and energy models are presented in Section II. Section III discusses related work. The proposed approach with the local search algorithms are presented in Section IV. Experimental results are given in Section V. Section VI concludes the paper.

II. PROBLEM DESCRIPTION

The target execution support considered in this work is a distributed computing system made up of a set M of m heterogeneous processors. The processors have different processing speed providing different processing performance (i.e. MIPS Million Instruction Per Second). Each processor $m_j \in M$ is DVS-enabled; it can be operated on a set DVS_j of voltage and relative speed pairs. Each pair is a supply voltage v_k and its corresponding relative speed ms_k. We consider that processors consume energy while idling; that is, when a processor is idling it is assumed that the lowest voltage is supplied [8].

A precedence constrained parallel application is represented by a *Directed Acyclic Graph* (DAG). The DAG is defined as $G = (T, E)$, where T is a finite set of nodes (vertices) and E is a finite set of edges. The node $t_i \in T$ is associated with one task t_i of the modeled parallel application. Each task $t_i \in T$ has an associated basic execution time which is an independent value for each processor. The basic execution time of task t_i on processor m_j at maximum speed and voltage is denoted as p_{ij}.

Each edge $(t_{i_1}, t_{i_2}) \in E$ (with $t_{i_1}, t_{i_2} \in T$) is a precedence constraint between tasks and represents inter-task communications. The weight on any edge $(t_{i_1}, t_{i_2}) \in E$ stands for the communication time, denoted as $c_{t_{i_1} t_{i_2}}$. However, a communication cost is only required when two tasks are assigned to different processors. Therefore it is neglected when they are assigned to the same processor. For a given DAG the communication to computation ratio (CCR) is a measure that indicates wether a task graph is communication intensive, computation intensive or moderate. For a given task graph, it is computed by the average communication cost divided by the average computation cost on a target system.

The energy model is based on previous finding and derived from the power consumption model in digital complementary metal-oxide semiconductor (CMOS) logic circuitry. It is directly related to frequency and supply voltage, and it is defined as [8]:

$$P_c = AC_{eff}V^2 f, \tag{1}$$

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 processors in this paper is defined as:

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

where $M[i]$ represents a vector containing the processor m_j where task t_i is allocated, V_i is the supply voltage of the processor m_j. On the other hand, the energy consumption during idle time is defined as:

$$E_i = \sum_{j=1}^{m} \sum_{idle_{jk} \in IDLES_j} AC_{eff}Vmin_j^2 I_{jk}, \tag{3}$$

where $IDLES_j$ is the set of idling slots on processor m_j, V_{minj} 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:

$$E_t = E_c + E_i. \tag{4}$$

The scheduling problem is the process of allocating the set T of t tasks to the set R of m DVS-enabled and unrelated processors, and assigning a starting time for each task to minimize the makespan and energy consumption. The makespan C_{max} of a schedule is its total *execution time*. The first task starts its execution at time 0, so that the makespan of a schedule is the maximum time at which one of the processors finishes its computation.

III. RELATED WORK

A well known scheduling algorithm in heterogeneous distributed computing systems for DAGs applications is the Heterogeneous Earliest Finish Time (HEFT) algorithm [9]. HEFT is founded on the list scheduling principle. It maintains

a list of all tasks of a given graph according to their priorities. It consists in two phases. In the first phase of the algorithm a ready task is selected from the priority list. The task with the highest priority is chosen. Then, a suitable processor that minimizes a predefined cost function is selected. HEFT is competitive in that it generates a schedule length comparable to other scheduling algorithms, with a low time complexity $(O(nlogn + (e + n)m))$.

An important number of energy conscious scheduling algorithms have been proposed in the literature [2], [5], [7], [8], [10]–[15]. These algorithms mainly differ on the assumptions they consider. However, the most common technique they exploit is DVS. Some energy-aware scheduling algorithms are based on the best-effort idea. Baskiyar and Palli [14] introduces an algorithm that combines Decisive Path Scheduling (DPS) with DVS to minimize both finish time and energy consumption. However, energy consumption due to idle time is not considered in the model. Wang et al. [4] proposed a best-effort scheduling algorithm to optimize energy. DVS is applied to no-critical tasks. A task is critical if it is in the critical path of the schedule. The proposed algorithm also considers reducing voltage during the communication phases between parallel tasks. Nevertheless, a loss of performance it's tolerated based on quality of service agreement to reduce more energy consumption. In this paper, we do not allow performance deterioration. Pecero et al. [5] use HEFT as a best-effort scheduling algorithm to compute schedules with good makespan at maximum voltage. Hence, an iterative local search algorithm is applied to reduce the energy consumption by scaling down the processor voltages to a proper DVS level without increasing makespan. This approach is close to the proposed in this paper, however the model does not consider energy consumption due to idle time. Rizvandi et al. report in [6] a heuristic called maximum minimum frequency DVFS. The proposed approach operates in two phases. The goal of the first phase is to find a schedule that minimizes the makespan. The second phase use a linear combination of the minimum and maximum processor frequencies to optimize energy without allowing makespan increase. Homogeneous processors with different voltage and frequency were considered. In this paper we consider heterogeneous systems. Lee and Zomaya in [8] propose heuristics that are designed with relative superiority as a novel objective function. The proposed algorithms consist in two phases. In the first phase a linear combination of both objectives is used by the heuristic algorithms as a best-effort approach. Then, a local search algorithm called $MCER$, which is a technique used to lowering energy consumption, although the technique may not scape from local optima. However, MCER is makespan conservative in that changes it makes (to the schedules generated by scheduling phase) are only validated if they do not increase schedule length. For each task in an application, MCER considers all of the other combinations of tasks, processors and voltage-speed pairs to check whether any of these combinations reduces energy consumption of the task without deteriorating the makespan.

IV. PROPOSED APPROACH

The solution provided in this work follows the independent approach. For the first phase of the algorithm we look for solutions with near-optimal makespan, for that we use HEFT as a best-effort scheduling algorithm. We consider HEFT because it is simple, well-known and competitive. Let us remark that HEFT uses the maximum voltage to compute the makespan. Once HEFT generates the schedule, we fix the makespan as a constraint and we optimize the energy consumption by a local search algorithm. Local search was one of the early techniques for combinatorial optimization. The principle is to refine a given initial solution point in the solution space by searching through a neighborhood of the solution point.

We introduce two new local search algorithms that adopt DVS to minimize energy consumption, however makespan is improved by allocating tasks to different current allocation (i.e., processor). **Algorithm 1** and **Algorithm 2** show the algorithm outline for the proposed local search. Both algorithms are intensive search process that follow the same framework, and are based on the best improvement iterative local search process, however the main difference relies on the way the neighbor of a given solution is defined and constructed. As that, the search space is differently explored. The Algorithm 1 is called $Best\_RT\_MVk$ for best improvement, and the notion of neighbor schedule is defined by picking a random task from T then transferring that task from current processor to another, for all processors in M and voltage - speed pairs (Vk). Algorithm 2 is called $Best\_RMVk\_T$ for best improvement, and the neighborhood is defined by randomly choosing a processor and a voltage-speed pair that corresponds to voltage level of the selected processor, then all tasks are evaluated on this new selected allocation. Algorithm 1 starts by computing makespan C_{max}' (line 3) and energy E_t' (line 4) of current schedule (S_{HEFT}) generated by HEFT. Then, $Best\_RT\_MVk$ performs the search on a set of randomly chosen tasks t_i (line 8). Current allocation is intensively explored by moving task t_i from its current allocation to all the processors in M and for all pairs of voltage - speed (line 13 to line 15). At each iteration of the search process we compute a new schedule (line 16) and its energy (line 17) and current makespan and energy are updated if the new schedule is locally better than previous solution (line 18 to line 24). Line 25 indicates that t_i is backward to its old position. After task t_i is evaluated on all the processors and voltage-speed pairs, only the best movement is considered (line 28). This process is repeated MAX_STEPS, however it is restarted as long as the current solution is improved (line 23). Algorithm 2 follows the same search process with different local search direction.

V. EXPERIMENTS

We report experimental results to validate the new proposed approach. We compare the proposed approach against with Slack Reclamation. It is an extension of the work proposed

Algorithm 1 Local Search Function Best_RT_MVk.

function Best_RT_MVk(S_{HEFT})
  $S \leftarrow S_{HEFT}$ ▷ Let S = current schedule
  Compute current C_{max}'
  Compute current E_t'
  $searchstep \leftarrow 0$
  while searchstep $<$ MAX_STEPS **do**
    $searchstep++$
    Choose a random task t_i
    Let m' = the current allocation for task t_i
    Let v' = the current voltage for m'
    $best\_m_j \leftarrow m'$
    $best\_v_k \leftarrow v'$
    for $\forall m_j \in M$ **do**
      for $\forall(v_k, ms_k) \in DVS_{m_j}$ **do** ▷ Next step explores a neighbor of current solution for possible makespan and energy improvement
        Allocate t_i on m_j with voltage v_k and relative speed ms_k
        Compute energy E_t
        Recompute C_{max}
        if $C_{max} \leq C'_{max}$ and $E_t < E'_t$ **then**
          $C'_{max} \leftarrow C_{max}$
          $E'_t \leftarrow E_t$
          $best\_m_j \leftarrow m_j$
          $best\_v_k \leftarrow v_k$
          $searchstep \leftarrow 0$
        end if ▷ Next step reinitializes the search process
        Allocate t_i on m' with voltage v' and its relative speed
      end for
    end for▷ Next step assigns t_i to the best m_j with best v_k
    Allocate t_i on $best\_m_j$ with voltage $best\_v_k$ and its relative speed
  end while
end function

in [4] and adapted to the heterogenous case in [7]. Wang et al. [4] proposed a list scheduling algorithm with DVFS to optimize energy, called HDVFS in this paper. We also compare the proposed local search against MCER proposed in [8]. We name the algorithm HMCER for HEFT+MCER. We also implemented a random local search in which the local search direction is randomly selected [5]. We call the algorithm HRLS. In HRLS, if the initial solution point is improved, it moves to the refined solution point. Otherwise, another search direction is randomly selected. A simple neighborhood point of a schedule in the solution space is another schedule which is obtained by transferring a task from a processor to another processor with another operating voltage level.

Algorithm 2 Local Search Function Best_RMVk_T.

function Best_RMVk_T(S_{HEFT})
  $S \leftarrow S_{HEFT}$ ▷ Let S = current schedule
  Compute current C_{max}'
  Compute current E_t'
  $searchstep \leftarrow 0$
  while searchstep $<$ MAX_STEPS **do**
    $searchstep++$
    Let m' = the current allocation for task t_i
    Let v' = the current voltage for m'
    $best\_m_j \leftarrow m'$
    $best\_v_k \leftarrow v'$
    Choose a random processor m_j
    Choose a random voltaje $v_k \in DVS_{m_j}$
    for $\forall t_i \in T$ **do** ▷ Next step explores a neighbor of current solution
      Allocate t_i on m_j with voltage v_k and relative speed ms_k
      Compute energy E_t
      Recompute C_{max}
      if $C_{max} \leq C'_{max}$ and $E_t < E'_t$ **then**
        $C'_{max} \leftarrow C_{max}$
        $E'_t \leftarrow E_t$
        $best\_t_i \leftarrow t_i$
        $best\_m_j \leftarrow m_j$
        $best\_v_k \leftarrow v_k$
        $searchstep \leftarrow 0$
      end if ▷ Next step reinitializes the search process
      Allocate t_i on m' with voltage v' and its relative speed
    end for▷ Next step assigns t_i to the best m_j with best v_k
    Allocate t_i on $best\_m_j$ with voltage $best\_v_k$ and its relative speed
  end while
end function

A. Experimental Settings

To evaluate performance of the proposed approach we have considered the following scenario. processor environment is heterogeneous in two aspects. First of them is generation of execution time, which is done independently for each task and processor. The second dimension concerns DVS: the DVS types are allocated by round-robin rule to each processors using six sets of DVS characteristics presented in Table I.

We have executed the algorithms on a set of structured real-world parallel applications. The four applications used for our experiments are the robot control application, a sparse matrix solver, fpppp problem from the *Standard Task Graph Set* (STG) [17], and the Laser Interferometer Gravitational-Wave Observatory (LIGO) application [16]. Table II summarizes the main characteristics for these applications: instances size, edges amount and the ratio between tasks and edges (ETR).

TABLE I
VOLTAGE-RELATIVE SPEED PAIRS [6], [8]

Level	Pair 1		Pair 2		Pair 3		Pair 4		Pair 5		Pair 6	
	Volt. (v_k)	Rel. Speed (%)	Volt. (v_k)	Rel. Speed (%)	Volt. (v_k)	Rel. Speed (%)	Volt. (v_k)	Rel. Speed (%)	Volt. (v_k)	Rel. Speed (%)	Volt. (v_k)	Rel. Speed (%)
0	1.50	100	2.20	100	1.50	100	1.75	100	1.20	100	1.35	100
1	1.20	80	1.90	85	1.40	90	1.40	80	1.15	90	1.25	85.7
2	0.90	50	1.60	65	1.30	80	1.20	60	1.10	80	1.20	71.5
3			1.30	50	1.20	70	0.90	40	1.05	70	1.10	57.1
4			1.00	35	1.10	60			1.00	60	0.9	32.2
5					1.00	50			0.90	50		
6					0.90	40						

ETR gives information on the degree of parallelism.

TABLE II
INSTANCE TYPES: TASKS AND EDGES NUMBERS, AND EDGE TASK RATIO.

Type	Tasks	Edges	ETR
LIGO	76	132	1.73
Robot	88	131	1.48
Sparse	96	67	0.69
Fpppp	334	1145	3.42

The STG set is composed of homogenous instances with constant execution time among processors. Since we are interested in heterogenous instances, we have only considered the structure of these applications and we have implemented the procedure described in [9] to consider heterogeneity. We fixed the parameter β to 1. Parameter β is basically the heterogeneity factor for processor speeds. A high percentage value (i.e., a percentage of 1) causes a significant difference in a task's computation cost among the processors. For each graph we have varied the CCR ratio. A randomization procedure which changes weight of edges was executed to assure the needed CCR. We have generated five CCRs (0.1, 0.5, 1, 5, 10) for each graph. Tested system sizes were 8, 16 and 32 processors. We have executed each algorithm 15 times on each instance. MAX_STEPS is fixed to 100 on the proposed algorithms.

B. Results

We have first evaluated the algorithms by Friedman test [18]. The Friedman test is a nonparametric statistical tool that allows to compare a set of a non normalized population to verify if exists significant statistical differences in the sample, additionally, the ranking shows which algorithm has the higher performance.

Tables III and IV depict the results obtained after applying the Friedman test, the ranking shows the performance of the algorithms, as this is a minimization problem, the higher the value of the ranking, the better the algorithm performs in finding a solution for the instance set.

Table III shows that the algorithm that outperforms related approaches when taking the makespan as a parameter for comparison is the $Best\_RMVk\_T$ algorithm, and the second one is HRLS. On the other hand, when the parameter taken into account is the energy consumption, the algorithm that seems to have the highest performance is $Best\_RT\_MVk$ followed by $Best\_RMVk\_T$, as shown in Table IV.

TABLE III
AVERAGE RANKINGS OF THE ALGORITHMS ON ALL THE SET OF INSTANCES BASED ON MAKESPAN

Algorithm	LIGO	Robot	Sparse	Fpppp
HDVFS	2.3066	2.1333	2.6800	2.1600
HMCER	3.6466	3.4400	3.4200	3.7400
HRLS	4.3533	4.55335	4.2266	4.5800
$Best\_RT\_MVk$	3.7066	3.7733	3.7133	3.5133
$Best\_RMVk\_T$	**4.6800**	**4.9666**	**4.2800**	**4.8466**

TABLE IV
AVERAGE RANKINGS OF THE ALGORITHMS ON ALL THE SET OF INSTANCES BASED ON ENERGY

Algorithm	LIGO	Robot	Sparse	Fpppp
HDVFS	2.2933	3.1066	1.9733	1.9999
HMCER	4.0400	3.5466	4.1800	4.0133
HRLS	4.2133	4.4533	4.3266	4.1333
$Best\_RT\_MVk$	**4.7600**	**4.8533**	**4.9000**	4.4000
$Best\_RMVk\_T$	4.6266	4.0266	4.5933	**5.4533**

Friedman statistic considering reduction performance (distributed according to χ^2 with 5 degrees of freedom) is 151.596. P-value computed by Friedman Test: 6.905198635109855E-11.

As most of the evaluated algorithms are based on stochastic methods, we perform a statistical hypothesis testing to verify if the differences between the best algorithm are significant if the null hypothesis is rejected, or otherwise, if due to the random nature of the solutions, the obtained results may be considered statistically equivalent.

Tables V and VI present the statistical tests that take into account the p-value obtained between all the algorithms with the best performing one, with a level of significance of $\alpha = 0.05$, if the the observed value is larger than the $1 - \alpha$ quantile of this distribution, the null hypothesis is rejected. As mentioned, this indicates that at least the candidate algorithm gives better performance than at least one of the others.

Table V depicts the statistical hypothesis testing. It shows that $Best\_RMVk\_T$ outperforms almost all the algorithms, but is statistically equivalent to algorithm $HRLS$, when the

parameter of comparison is Makespan. Meanwhile, when the algorithms are compared by energy, $Best\_RT\_MVk$ outperforms HDVFS, and is statistically equivalent to HRLS and in some sets of instances, to $Best\_RMVk\_T$ and HMCER.

TABLE V
STATISTICAL HYPOTHESIS TESTING FOR THE BEST ALGORITHM BY MAKESPAN WITH $\alpha = 0.05$

Set	Algorithm vs. $Best\_RMVk\_T$	p-value	H_0 Rejected
LIGO	HDVFS	7.9385E-15	Yes
	HMCER	7.1861E-4	Yes
	$Best\_RT\_MVk$	0.0014	Yes
	HRLS	0.2849	No
Robot	HDVFS	1.7885E-20	Yes
	HMCER	5.8171E-7	Yes
	$Best\_RT\_MVk$	9.3797E-5	Yes
	HRLS	0.1760	No
Sparse	HDVFS	1.6300E-7	Yes
	HMCER	0.0048	Yes
	$Best\_RT\_MVk$	0.06361	No
	HRLS	0.8614	No
Fpppp	HDVFS	1.4409E-18	Yes
	HMCER	2.9186E-4	Yes
	$Best\_RT\_MVk$	1.2749E-5	Yes
	HRLS	0.3827	No

TABLE VI
STATISTICAL HYPOTHESIS TESTING FOR THE BEST ALGORITHM BY ENERGY WITH $\alpha = 0.05$

Set	Algorithm vs. $Best\_RT\_MVk$	p-value	H_0 Rejected
LIGO	HDVFS	6.7997E-16	Yes
	$Best\_RMVk\_T$	0.6625	No
	HMCER	0.0184	Yes
	HRLS	0.0735	No
Robot	HDVFS	1.0822E-8	Yes
	$Best\_RMVk\_T$	0.0068	Yes
	HMCER	1.8936E-5	Yes
	HRLS	0.1904	No
Sparse	HDVFS	9.7265E-22	Yes
	$Best\_RMVk\_T$	0.3154	No
	HMCER	0.0184	Yes
	HRLS	0.0605	No
Fpppp	HDVFS	3.9708E-15	Yes
	$Best\_RMVk\_T$	5.6508E-4	Yes
	HMCER	0.2056	No
	HRLS	0.3827	No

Tables VII, VIII, and IX display performance results for the simulations. These tables respectively show the gain computed by all the compared algorithms according to number of processores, CCR and by application. All the algorithms are compared against HDVFS. These results show that the new proposed local search algorithms improve on average the results obtained by HDVFS in both objectives. The improvement in makespan by $Best\_RMVk\_T$ is 4.41% and 15.58% when energy minimization is considered. The gain in the makespan objective by $Best\_RT\_MVk$ is 1.25% and 13.3% on energy consumption. In both cases, the proposed algorithms are able to reduce energy without degradation on makespan. We can observe that the factor that impact the most on the behavior of most of the compared algorithms is the number of processors. We suspect that this behavior is due to the idle time considered in the energy consumption model. The solutions provided by the algorithms do not use all the processors, hence energy consumption due to idle time is important. In this case, we consider that a better approach is switching-off the idle processors.

VI. CONCLUDING REMARKS

We have investigated the problem of scheduling applications in heterogeneous distributed computing systems considering energy issue. We have presented an approach based on the best-effort idea and introduced two local search algorithms ($Best\_RT\_MVk$ and $Best\_RMVk\_T$). The local search algorithms promote the use of DVS to optimize energy. The set of simulation results shows that the proposed approach is able to optimize makespan and has a good performance behavior when optimizing energy. Interestingly, these results also reveal that energy consumption due to idle time is important and a better approach can be switching off idle processors, however the latencies due to turning off should be considered. As a future work, we plan to extend our model to consider switching on-off model and DVS. We also consider virtualization to maximize resource utilization for the used processors. Another important prospect is to investigate multi-objective local search algorithms.

ACKNOWLEDGMENT

This work is supported by the National Research Fund (FNR) of Luxembourg through project Green-IT no. C09/IS/05. We also would like to thank CONACyT supporting researchers from ITCM.

REFERENCES

[1] DatacenterDynamics: Datacenterdynamics research report, 2012. http://www.datacenterdynamics.com/research (Consulted Online 2012).

[2] G. Valentini, W. Lassonde, S. Khan, N. Min-Allah, S. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Zomaya, C. Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," Cluster Computing, on-line first, DOI: 10.1007/s10586-011-0171-x, pp. 1–13.

[3] A. Andrei, P. Eles, Z. Peng, M. Schmitz, and B. Al-Hashimi, "Energy Optimization of Multiprocessor Systems on Chip by Voltage Selection," IEEE Trans on Very Large Scale Integration Systems, Vol. 15, Issue 3, March, 2007, pp. 262–275.

[4] L. Wang, G. von Laszewski, and J. Dayal, "Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS," IEEE/ACM CCGRID'10, Australia, 2010, pp. 368–377.

[5] J. E. Pecero, P. Bouvry, C. J. Barrios Hernandez, "Low Energy and High Performance Scheduling on Scalable Computing Systems," CLCAR'10, ISBN: 978-85-7727-252-5, Gramado, RS, Brazil, 2010, pp. 1–8.

[6] N. B. Rizvandi, J. Taheri, A. Y. Zomaya, "Some observations on optimal frequency selection in dvfs-based energy consumption minimization," J. Parallel Distr Com, vol. 71 no. 8, 2011, pp. 1154–1164.

[7] J. E. Pecero, P. Bouvry, H. J. Fraire Huacuja, and S. U. Khan, "A Multi-objective GRASP Algorithm for Joint Optimization of Energy Consumption and Schedule Length of Precedence-Constrained Applications," IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), 2011 DO: 10.1109/DASC.2011.97, pp. 510–517.

TABLE VII
COMPARATIVE RESULTS BY NUMBER OF PROCESSORS

No. of Procs	HDVFS over HMCER (%)		HDVFS over HRLS (%)		HDVFS over $Best\_RT\_MVk$ (%)		HDVFS over $Best\_RMVk\_T$ (%)	
	Makespan	Energy	Makespan	Energy	Makespan	Energy	Makespan	Energy
8	0	14	1	12	0	14	2	19
16	0	13	3	13	1	14	6	17
32	4	11	5	11	4	12	9	14

TABLE VIII
COMPARATIVE RESULTS BY CCR

CCR	HDVFS over HMCER (%)		HDVFS over HRLS (%)		HDVFS over $Best\_RT\_MVk$(%)		HDVFS over $Best\_RMVk\_T$ (%)	
	Makespan	Energy	Makespan	Energy	Makespan	Energy	Makespan	Energy
0.1	0	12	0	12	0	12	0	15
0.5	0	17	0	17	0	17	0	19
1	0	9	0	10	0	10	1	11
5	0	11	2	11	1	12	3	13
10	3	9	5	9	4	11	9	14

TABLE IX
COMPARATIVE RESULTS BY APPLICATIONS

Appl.	HDVFS over HMCER (%)		HDVFS over HRLS (%)		HDVFS over $Best\_RT\_MVk$ (%)		HDVFS over $Best\_RMVk\_T$ (%)	
	Makespan	Energy	Makespan	Energy	Makespan	Energy	Makespan	Energy
LIGO	4	9	6	11	5	11	7	13
Robot	0	0	2	1	1	2	3	2
Sparse	1	28	4	30	3	32	10	33
Fpppp	0	13	1	12	1	13	3	17

[8] Y.C. Lee, and A. Y. Zomaya, "Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 8, Aug. 2011, pp. 1374–1381.

[9] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and Low Complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, Issue 3, 2002, pp. 260–274.

[10] 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. 59, no. 1, 2012, pp. 323–360.

[11] S. U. 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, 2009, pp. 346–360.

[12] J. K Kim, H. J. Siegel, A. A. Maciejewski, R. Eigenmann, "Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling," IEEE Trans. on Parallel Distr. Syst., vol. 19, 2008, pp. 1445–1457.

[13] S. Albers, "Energy-efficient algorithms," Commun. ACM, vol. 53, 2010, pp. 86–96.

[14] S. Baskiyar, and R. Abdel-Kader, "Energy aware DAG scheduling on heterogeneous systems," Cluster Computing, vol. 13, 2010, pp. 373–383.

[15] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, and A. Y. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," J. Parallel Distr Com, vol. 71, no. 11, 2011, pp. 1497–1508.

[16] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, and J. McNabb, A case study on the use of workflow technologies for scientific analysis: Gravitational Wave Data Analysis, In Taylor, I., D. Gannon, and M. Shields (eds.), *Workflows for e-Science Scientific Workflows for Grids*, Springer, 2007, pp. 39–59.

[17] T. Tobita, and H. Kasahara, "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms", J. Scheduling, vol. 5, Issue 5, 2002, pp. 379–394.

[18] P. H. Kvam and B. Vidakovic, *Nonparametric Statistics with Applications to Science and Engineering*, Wiley Series in Probability and Statistics; Wiley-Interscience, 2007.