Energy Consumption Reduction with DVFS for Message Passing Iterative Applications on Heterogeneous Architectures

Jean-Claude Charr, Raphaël Couturier, Ahmed Fanfakh and Arnaud Giersch
FEMTO-ST Institute, University of Franche-Comté
IUT de Belfort-Montbéliard, 19 avenue du Maréchal Juin, BP 527, 90016 Belfort cedex, France
Email: {jean-claude.charr,raphael.couturier,ahmed.fanfakh_badri_muslim,arnaud.giersch}@univ-fcomte.fr

***Abstract*—Computing platforms are consuming more and more energy due to the increasing number of nodes composing them. To minimize the operating costs of these platforms many techniques have been used. Dynamic voltage and frequency scaling (DVFS) is one of them. It reduces the frequency of a CPU to lower its energy consumption. However, lowering the frequency of a CPU may increase the execution time of an application running on that processor. Therefore, the frequency that gives the best trade-off between the energy consumption and the performance of an application must be selected.**

In this paper, a new online frequency selecting algorithm for heterogeneous platforms (heterogeneous CPUs) is presented. It selects the frequencies and tries to give the best trade-off between energy saving and performance degradation, for each node computing the message passing iterative application. The algorithm has a small overhead and works without training or profiling. It uses a new energy model for message passing iterative applications running on a heterogeneous platform. The proposed algorithm is evaluated on the SimGrid simulator while running the NAS parallel benchmarks. The experiments show that it reduces the energy consumption by up to 34 % while limiting the performance degradation as much as possible. Finally, the algorithm is compared to an existing method, the comparison results show that it outperforms the latter, on average it saves 4 % more energy while keeping the same performance.

I. Introduction

The need for more computing power is continually increasing. To partially satisfy this need, most supercomputers constructors just put more computing nodes in their platform. The resulting platforms may achieve higher floating point operations per second (FLOPS), but the energy consumption and the heat dissipation are also increased. As an example, the Chinese supercomputer Tianhe-2 had the highest FLOPS in November 2014 according to the Top500 list [1]. However, it was also the most power hungry platform with its over 3 million cores consuming around 17.8 megawatts. Moreover, according to the U.S. annual energy outlook 2014 [2], the price of energy for 1 megawatt-hour was approximately equal to $70. Therefore, the price of the energy consumed by the Tianhe-2 platform is approximately more than $10 million each year. The computing platforms must be more energy efficient and offer the highest number of FLOPS per watt possible, such as the L-CSC from the GSI Helmholtz Center which became the top of the Green500 list in November 2014 [3]. This heterogeneous platform executes more than 5 GFLOPS per watt while consuming 57.15 kilowatts.

Besides platform improvements, there are many software and hardware techniques to lower the energy consumption of these platforms, such as scheduling, DVFS, ... DVFS is a widely used process to reduce the energy consumption of a processor by lowering its frequency [4]. However, it also reduces the number of FLOPS executed by the processor which may increase the execution time of the application running over that processor. Therefore, researchers use different optimization strategies to select the frequency that gives the best trade-off between the energy reduction and performance degradation ratio. In [5], a frequency selecting algorithm was proposed to reduce the energy consumption of message passing iterative applications running over homogeneous platforms. The results of the experiments show significant energy consumption reductions. In this paper, a new frequency selecting algorithm adapted for heterogeneous platform is presented. It selects the vector of frequencies, for a heterogeneous platform running a message passing iterative application, that simultaneously tries to offer the maximum energy reduction and minimum performance degradation ratio. The algorithm has a very small overhead, works online and does not need any training or profiling.

This paper is organized as follows: Section II presents some related works from other authors. Section III describes how the execution time of message passing programs can be predicted. It also presents an energy model that predicts the energy consumption of an application running over a heterogeneous platform. Section IV presents the energy-performance objective function that maximizes the reduction of energy consumption while minimizing the degradation of the program's performance. Section V details the proposed frequency selecting algorithm then the precision of the proposed algorithm is verified. Section VI presents the results of applying the algorithm on the NAS parallel benchmarks and executing them on a heterogeneous platform. It shows the results of running three different power scenarios and comparing them. Moreover, it also shows the comparison results between the proposed method and an existing method. Finally, in Section VII the paper ends with a summary and some future works.

II. Related works

DVFS is a technique used in modern processors to scale down both the voltage and the frequency of the CPU while computing, in order to reduce the energy consumption of

the processor. DVFS is also allowed in GPUs to achieve the same goal. Reducing the frequency of a processor lowers its number of FLOPS and may degrade the performance of the application running on that processor, especially if it is compute bound. Therefore selecting the appropriate frequency for a processor to satisfy some objectives, while taking into account all the constraints, is not a trivial operation. Many researchers used different strategies to tackle this problem. Some of them developed online methods that compute the new frequency while executing the application, such as [6], [7]. Others used offline methods that may need to run the application and profile it before selecting the new frequency, such as [8], [9]. The methods could be heuristics, exact or brute force methods that satisfy varied objectives such as energy reduction or performance. They also could be adapted to the execution's environment and the type of the application such as sequential, parallel or distributed architecture, homogeneous or heterogeneous platform, synchronous or asynchronous application, ...

In this paper, we are interested in reducing energy for message passing iterative synchronous applications running over heterogeneous platforms. Some works have already been done for such platforms and they can be classified into two types of heterogeneous platforms:

- the platform is composed of homogeneous GPUs and homogeneous CPUs.
- the platform is only composed of heterogeneous CPUs.

For the first type of platform, the computing intensive parallel tasks are executed on the GPUs and the rest are executed on the CPUs. Luley et al. [10], proposed a heterogeneous cluster composed of Intel Xeon CPUs and NVIDIA GPUs. Their main goal was to maximize the energy efficiency of the platform during computation by maximizing the number of FLOPS per watt generated. In [11], Kai Ma et al. developed a scheduling algorithm that distributes workloads proportional to the computing power of the nodes which could be a GPU or a CPU. All the tasks must be completed at the same time. In [12], Rong et al. showed that a heterogeneous (GPUs and CPUs) cluster that enables DVFS gave better energy and performance efficiency than other clusters only composed of CPUs.

The work presented in this paper concerns the second type of platform, with heterogeneous CPUs. Many methods were conceived to reduce the energy consumption of this type of platform. Naveen et al. [13] developed a method that minimizes the value of $energy \times delay^2$ (the delay is the sum of slack times that happen during synchronous communications) by dynamically assigning new frequencies to the CPUs of the heterogeneous cluster. Lizhe et al. [14] proposed an algorithm that divides the executed tasks into two types: the critical and non critical tasks. The algorithm scales down the frequency of non critical tasks proportionally to their slack and communication times while limiting the performance degradation percentage to less than 10 %. In [15], they developed a heterogeneous cluster composed of two types of Intel and AMD processors. They use a gradient method to predict the impact of DVFS operations on performance. In [16] and [17], the best frequencies for a specified heterogeneous cluster are selected offline using some heuristic. Chen et al. [18] used a greedy dynamic programming approach to minimize the power consumption of heterogeneous servers while respecting given time constraints. This approach had considerable overhead. In contrast to the above described papers, this paper presents the following contributions :

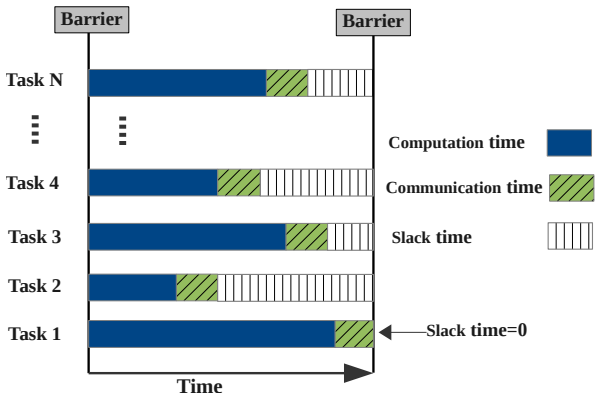


Figure 1: Parallel tasks on a heterogeneous platform

1) two new energy and performance models for message passing iterative synchronous applications running over a heterogeneous platform. Both models take into account communication and slack times. The models can predict the required energy and the execution time of the application.
2) a new online frequency selecting algorithm for heterogeneous platforms. The algorithm has a very small overhead and does not need any training or profiling. It uses a new optimization function which simultaneously maximizes the performance and minimizes the energy consumption of a message passing iterative synchronous application.

III. The performance and energy consumption measurements on heterogeneous architecture

A. The execution time of message passing distributed iterative applications on a heterogeneous platform

In this paper, we are interested in reducing the energy consumption of message passing distributed iterative synchronous applications running over heterogeneous platforms. A heterogeneous platform is defined as a collection of heterogeneous computing nodes interconnected via a high speed homogeneous network. Therefore, each node has different characteristics such as computing power (FLOPS), energy consumption, CPU's frequency range, ... but they all have the same network bandwidth and latency.

The overall execution time of a distributed iterative synchronous application over a heterogeneous platform consists of the sum of the computation time and the communication time for every iteration on a node. However, due to the heterogeneous computation power of the computing nodes, slack times may occur when fast nodes have to wait, during synchronous communications, for the slower nodes to finish their computations (see Figure 1). Therefore, the overall exe-

cution time of the program is the execution time of the slowest task which has the highest computation time and no slack time.

Dynamic Voltage and Frequency Scaling (DVFS) is a process, implemented in modern processors, that reduces the energy consumption of a CPU by scaling down its voltage and frequency. Since DVFS lowers the frequency of a CPU and consequently its computing power, the execution time of a program running over that scaled down processor may increase, especially if the program is compute bound. The frequency reduction process can be expressed by the scaling factor S which is the ratio between the maximum and the new frequency of a CPU as in (1).

$$S = \frac{F_{max}}{F_{new}} \tag{1}$$

The execution time of a compute bound sequential program is linearly proportional to the frequency scaling factor S. On the other hand, message passing distributed applications consist of two parts: computation and communication. The execution time of the computation part is linearly proportional to the frequency scaling factor S but the communication time is not affected by the scaling factor because the processors involved remain idle during the communications [19]. The communication time for a task is the summation of periods of time that begin with an MPI call for sending or receiving a message until the message is synchronously sent or received.

Since in a heterogeneous platform each node has different characteristics, especially different frequency gears, when applying DVFS operations on these nodes, they may get different scaling factors represented by a scaling vector: $(S_1, S_2, \ldots, S_N)$ where S_i is the scaling factor of processor i. To be able to predict the execution time of message passing synchronous iterative applications running over a heterogeneous platform, for different vectors of scaling factors, the communication time and the computation time for all the tasks must be measured during the first iteration before applying any DVFS operation. Then the execution time for one iteration of the application with any vector of scaling factors can be predicted using (2).

$$T_{New} = \max_{i=1,2,\ldots,N}(T_{cpOld_i} \cdot S_i) + MinT_{cm} \tag{2}$$

Where:

$$MinT_{cm} = \min_{i=1,2,\ldots,N}(T_{cm_i}) \tag{3}$$

where T_{cpOld_i} is the computation time of processor i during the first iteration and $MinT_{cm}$ is the communication time of the slowest processor from the first iteration. The model computes the maximum computation time with scaling factor from each node added to the communication time of the slowest node. It means only the communication time without any slack time is taken into account. Therefore, the execution time of the iterative application is equal to the execution time of one iteration as in (2) multiplied by the number of iterations of that application.

This prediction model is developed from the model to predict the execution time of message passing distributed applications for homogeneous architectures [5]. The execution time prediction model is used in the method to optimize both the energy consumption and the performance of iterative methods, which is presented in the following sections.

B. Energy model for heterogeneous platform

Many researchers [20], [21], [22], [4] divide the power consumed by a processor into two power metrics: the static and the dynamic power. While the first one is consumed as long as the computing unit is turned on, the latter is only consumed during computation times. The dynamic power P_d is related to the switching activity α, load capacitance C_L, the supply voltage V and operational frequency F, as shown in (4).

$$P_d = \alpha \cdot C_L \cdot V^2 \cdot F \tag{4}$$

The static power P_s captures the leakage power as follows:

$$P_s = V \cdot N_{trans} \cdot K_{design} \cdot I_{leak} \tag{5}$$

where V is the supply voltage, N_{trans} is the number of transistors, K_{design} is a design dependent parameter and I_{leak} is a technology dependent parameter. The energy consumed by an individual processor to execute a given program can be computed as:

$$E_{ind} = P_d \cdot T_{cp} + P_s \cdot T \tag{6}$$

where T is the execution time of the program, T_{cp} is the computation time and $T_{cp} \leq T$. T_{cp} may be equal to T if there is no communication and no slack time.

The main objective of DVFS operation is to reduce the overall energy consumption [**?**]. The operational frequency F depends linearly on the supply voltage V, i.e., $V = \beta \cdot F$ with some constant β. This equation is used to study the change of the dynamic voltage with respect to various frequency values in [21]. The reduction process of the frequency can be expressed by the scaling factor S which is the ratio between the maximum and the new frequency as in (1). The CPU governors are power schemes supplied by the operating system's kernel to lower a core's frequency. The new frequency F_{new} from (1) can be calculated as follows:

$$F_{new} = S^{-1} \cdot F_{max} \tag{7}$$

Replacing F_{new} in (4) as in (7) gives the following equation for dynamic power consumption:

$$\begin{aligned} P_{dNew} &= \alpha \cdot C_L \cdot V^2 \cdot F_{new} = \alpha \cdot C_L \cdot \beta^2 \cdot {F_{new}}^3 \\ &= \alpha \cdot C_L \cdot V^2 \cdot F_{max} \cdot S^{-3} = P_{dOld} \cdot S^{-3} \end{aligned} \tag{8}$$

where P_{dNew} and P_{dOld} are the dynamic power consumed with the new frequency and the maximum frequency respectively.

According to (8) the dynamic power is reduced by a factor of S^{-3} when reducing the frequency by a factor of S [21]. Since the FLOPS of a CPU is proportional to the frequency of a CPU, the computation time is increased proportionally to S. The new dynamic energy is the dynamic power multiplied by the new time of computation and is given by the following equation:

$$E_{dNew} = P_{dOld} \cdot S^{-3} \cdot (T_{cp} \cdot S) = S^{-2} \cdot P_{dOld} \cdot T_{cp} \tag{9}$$

The static power is related to the power leakage of the CPU and is consumed during computation and even when idle. As in [21], [22], the static power of a processor is considered as constant during idle and computation periods, and for all its available frequencies. The static energy is the static power

multiplied by the execution time of the program. According to the execution time model in (2), the execution time of the program is the sum of the computation and the communication times. The computation time is linearly related to the frequency scaling factor, while this scaling factor does not affect the communication time. The static energy of a processor after scaling its frequency is computed as follows:

$$E_S = P_s \cdot (T_{cp} \cdot S + T_{cm}) \tag{10}$$

In the considered heterogeneous platform, each processor i may have different dynamic and static powers, noted as P_{d_i} and P_{s_i} respectively. Therefore, even if the distributed message passing iterative application is load balanced, the computation time of each CPU i noted T_{cp_i} may be different and different frequency scaling factors may be computed in order to decrease the overall energy consumption of the application and reduce slack times. The communication time of a processor i is noted as T_{cm_i} and could contain slack times when communicating with slower nodes, see Figure 1. Therefore, all nodes do not have equal communication times. While the dynamic energy is computed according to the frequency scaling factor and the dynamic power of each node as in (9), the static energy is computed as the sum of the execution time of one iteration multiplied by the static power of each processor. The overall energy consumption of a message passing distributed application executed over a heterogeneous platform during one iteration is the summation of all dynamic and static energies for each processor. It is computed as follows:

$$E = \sum_{i=1}^{N} (S_i^{-2} \cdot P_{d_i} \cdot T_{cp_i}) + \sum_{i=1}^{N} (P_{s_i} \cdot (\max_{i=1,2,\ldots,N} (T_{cp_i} \cdot S_i) + MinT_{cm})) \tag{11}$$

Reducing the frequencies of the processors according to the vector of scaling factors $(S_1, S_2, \ldots, S_N)$ may degrade the performance of the application and thus, increase the static energy because the execution time is increased [23]. The overall energy consumption for the iterative application can be measured by measuring the energy consumption for one iteration as in (11) multiplied by the number of iterations of that application.

IV. Optimization of both energy consumption and performance

Using the lowest frequency for each processor does not necessarily give the most energy efficient execution of an application. Indeed, even though the dynamic power is reduced while scaling down the frequency of a processor, its computation power is proportionally decreased. Hence, the execution time might be drastically increased and during that time, dynamic and static powers are being consumed. Therefore, it might cancel any gains achieved by scaling down the frequency of all nodes to the minimum and the overall energy consumption of the application might not be the optimal one. It is not trivial to select the appropriate frequency scaling factor for each processor while considering the characteristics of each processor (computation power, range of frequencies, dynamic and static powers) and the task executed (computation/communication ratio). The aim being to reduce the overall energy consumption and to avoid increasing significantly the execution time. In our previous work [5], we proposed a method that selects the optimal frequency scaling factor for a homogeneous cluster executing a message passing iterative synchronous application while giving the best trade-off between the energy consumption and the performance for such applications. In this work we are interested in heterogeneous clusters as described above. Due to the heterogeneity of the processors, a vector of scaling factors should be selected and it must give the best trade-off between energy consumption and performance.

The relation between the energy consumption and the execution time for an application is complex and nonlinear, Thus, unlike the relation between the execution time and the scaling factor, the relation between the energy and the frequency scaling factors is nonlinear, for more details refer to [19]. Moreover, these relations are not measured using the same metric. To solve this problem, the execution time is normalized by computing the ratio between the new execution time (after scaling down the frequencies of some processors) and the initial one (with maximum frequency for all nodes) as follows:

$$P_{Norm} = \frac{T_{New}}{T_{Old}} = \frac{\max_{i=1,2,\ldots,N}(T_{cp_i} \cdot S_i) + MinT_{cm}}{\max_{i=1,2,\ldots,N}(T_{cp_i} + T_{cm_i})} \tag{12}$$

In the same way, the energy is normalized by computing the ratio between the consumed energy while scaling down the frequency and the consumed energy with maximum frequency for all nodes:

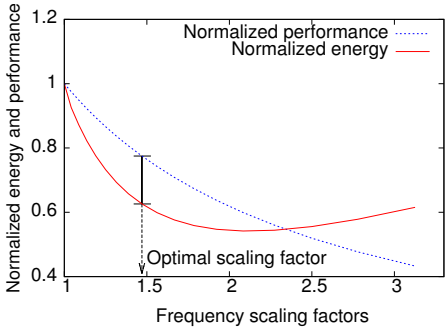
$$E_{Norm} = \frac{E_{Reduced}}{E_{Original}} = \frac{\sum_{i=1}^{N}(S_i^{-2} \cdot P_{d_i} \cdot T_{cp_i}) + \sum_{i=1}^{N}(P_{s_i} \cdot T_{New})}{\sum_{i=1}^{N}(P_{d_i} \cdot T_{cp_i}) + \sum_{i=1}^{N}(P_{s_i} \cdot T_{Old})} \tag{13}$$

Where $E_{Reduced}$ and $E_{Original}$ are computed using (11) and T_{New} and T_{Old} are computed as in (12).

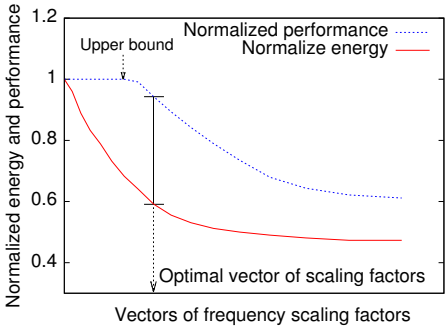
While the main goal is to optimize the energy and execution time at the same time, the normalized energy and execution time curves do not evolve (increase/decrease) in the same way. According to the equations (12) and (13), the vector of frequency scaling factors $S_1, S_2, \ldots, S_N$ reduce both the energy and the execution time simultaneously. But the main objective is to produce maximum energy reduction with minimum execution time reduction.

This problem can be solved by making the optimization process for energy and execution time follow the same evolution according to the vector of scaling factors. Therefore, the equation of the normalized execution time is inverted which gives the normalized performance equation, as follows:

$$P_{Norm} = \frac{T_{Old}}{T_{New}} = \frac{\max_{i=1,2,\ldots,N}(T_{cp_i} + T_{cm_i})}{\max_{i=1,2,\ldots,N}(T_{cp_i} \cdot S_i) + MinT_{cm}} \tag{14}$$



(a) Homogeneous platform



(b) Heterogeneous platform

Figure 2: The energy and performance relation

Then, the objective function can be modeled in order to find the maximum distance between the energy curve (13) and the performance curve (14) over all available sets of scaling factors. This represents the minimum energy consumption with minimum execution time (maximum performance) at the same time, see Figure 2a or Figure 2b. Then the objective function has the following form:

$$MaxDist = \max_{\substack{i=1,\dots F \\ j=1,\dots,N}} (\overbrace{P_{Norm}(S_{ij})}^{\text{Maximize}} - \overbrace{E_{Norm}(S_{ij})}^{\text{Minimize}}) \qquad (15)$$

where N is the number of nodes and F is the number of available frequencies for each node. Then, the optimal set of scaling factors that satisfies (15) can be selected. The objective function can work with any energy model or any power values for each node (static and dynamic powers). However, the most important energy reduction gain can be achieved when the energy curve has a convex form as shown in [22], [21], [6].

V. The Scaling Factors Selection Algorithm for Heterogeneous Platforms

A. The algorithm details

In this section, Algorithm 1 is presented. It selects the frequency scaling factors vector that gives the best trade-off between minimizing the energy consumption and maximizing the performance of a message passing synchronous iterative application executed on a heterogeneous platform. It works online during the execution time of the iterative message passing program. It uses information gathered during the first iteration such as the computation time and the communication

Algorithm 1 frequency scaling factors selection algorithm

Require:

- T_{cp_i} array of all computation times for all nodes during one iteration and with highest frequency.
- T_{cm_i} array of all communication times for all nodes during one iteration and with highest frequency.
- F_{max_i} array of the maximum frequencies for all nodes.
- P_{d_i} array of the dynamic powers for all nodes.
- P_{s_i} array of the static powers for all nodes.
- F_{diff_i} array of the differences between two successive frequencies for all nodes.

Ensure: $S_{opt_1}, S_{opt_2} \dots, S_{opt_N}$ is a vector of optimal scaling factors

1: $S_{cp_i} \leftarrow \frac{\max_{i=1,2,\dots,N}(T_{cp_i})}{T_{cp_i}}$
2: $F_i \leftarrow \frac{F_{max_i}}{S_{cp_i}},\ i = 1, 2, \cdots, N$
3: Round the computed initial frequencies F_i to the closest one available in each node.
4: **if** (not the first frequency) **then**
5: $\quad F_i \leftarrow F_i + F_{diff_i},\ i = 1, \dots, N.$
6: **end if**
7: $T_{Old} \leftarrow \max_{i=1,\dots,N}(T_{cp_i} + T_{cm_i})$
8: $E_{Original} \leftarrow \sum_{i=1}^{N} (P_{d_i} \cdot T_{cp_i} + P_{s_i} \cdot T_{Old})$
9: $S_{opt_i} \leftarrow 1,\ i = 1, \dots, N.$
10: $Dist \leftarrow 0$
11: **while** (all nodes not reach their minimum frequency) **do**
12: $\quad$ **if** (not the last freq. **and** not the slowest node) **then**
13: $\quad\quad F_i \leftarrow F_i - F_{diff_i},\ i = 1, \dots, N.$
14: $\quad\quad S_i \leftarrow \frac{F_{max_i}}{F_i},\ i = 1, \dots, N.$
15: $\quad$ **end if**
16: $\quad T_{New} \leftarrow \max_{i=1,\dots,N}(T_{cp_i} \cdot S_i) + MinT_{cm}$
17: $\quad E_{Reduced} \leftarrow \sum_{i=1}^{N} (S_i^{-2} \cdot P_{d_i} \cdot T_{cp_i} + P_{s_i} \cdot T_{New})$
18: $\quad P_{Norm} \leftarrow \frac{T_{Old}}{T_{New}}$
19: $\quad E_{Norm} \leftarrow \frac{E_{Reduced}}{E_{Original}}$
20: $\quad$ **if** $(P_{Norm} - E_{Norm} > Dist)$ **then**
21: $\quad\quad S_{opt_i} \leftarrow S_i,\ i = 1, \dots, N.$
22: $\quad\quad Dist \leftarrow P_{Norm} - E_{Norm}$
23: $\quad$ **end if**
24: **end while**
25: Return $S_{opt_1}, S_{opt_2}, \dots, S_{opt_N}$

Algorithm 2 DVFS algorithm

1: **for** $k = 1$ to *some iterations* **do**
2: $\quad$ Computations section.
3: $\quad$ Communications section.
4: $\quad$ **if** $(k = 1)$ **then**
5: $\quad\quad$ Gather all times of computation and communication from each node.
6: $\quad\quad$ Call Algorithm 1.
7: $\quad\quad$ Compute the new frequencies from the returned optimal scaling factors.
8: $\quad\quad$ Set the new frequencies to nodes.
9: $\quad$ **end if**
10: **end for**

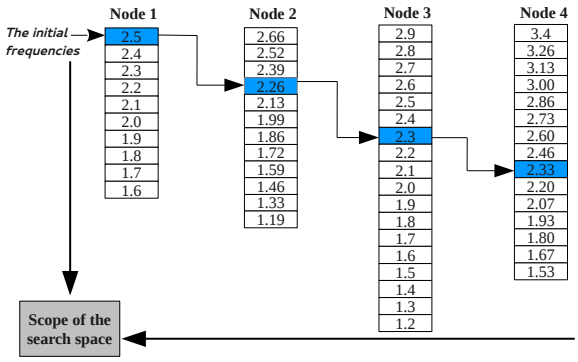


Figure 3: Selecting the initial frequencies

time in one iteration for each node. The algorithm is executed after the first iteration and returns a vector of optimal frequency scaling factors that satisfies the objective function (15). The program applies DVFS operations to change the frequencies of the CPUs according to the computed scaling factors. This algorithm is called just once during the execution of the program. Algorithm 2 shows where and when the proposed scaling algorithm is called in the iterative MPI program.

The nodes in a heterogeneous platform have different computing powers, thus while executing message passing iterative synchronous applications, fast nodes have to wait for the slower ones to finish their computations before being able to synchronously communicate with them as in Figure 1. These periods are called idle or slack times. The algorithm takes into account this problem and tries to reduce these slack times when selecting the frequency scaling factors vector. At first, it selects initial frequency scaling factors that increase the execution times of fast nodes and minimize the differences between the computation times of fast and slow nodes. The value of the initial frequency scaling factor for each node is inversely proportional to its computation time that was gathered from the first iteration. These initial frequency scaling factors are computed as a ratio between the computation time of the slowest node and the computation time of the node i as follows:

$$S_{cp_i} = \frac{\max_{i=1,2,\ldots,N}(T_{cp_i})}{T_{cp_i}} \tag{16}$$

Using the initial frequency scaling factors computed in (16), the algorithm computes the initial frequencies for all nodes as a ratio between the maximum frequency of node i and the computation scaling factor S_{cp_i} as follows:

$$F_i = \frac{F_{max_i}}{S_{cp_i}},\ i = 1, 2, \ldots, N \tag{17}$$

If the computed initial frequency for a node is not available in the gears of that node, it is replaced by the nearest available frequency. In Figure 3, the nodes are sorted by their computing power in ascending order and the frequencies of the faster nodes are scaled down according to the computed initial frequency scaling factors. The resulting new frequencies are highlighted in Figure 3. This set of frequencies can be considered as a higher bound for the search space of the optimal vector of frequencies because selecting scaling factors higher than the higher bound will not improve the performance of the application and it will increase its overall energy consumption. Therefore the algorithm that selects the frequency scaling factors starts the search method from these initial frequencies and takes a downward search direction toward lower frequencies. The algorithm iterates on all remaining frequencies, from the higher bound until all nodes reach their minimum frequencies, to compute their overall energy consumption and performance, and select the optimal frequency scaling factors vector. At each iteration the algorithm determines the slowest node according to the equation (2) and keeps its frequency unchanged, while it lowers the frequency of all other nodes by one gear. The new overall energy consumption and execution time are computed according to the new scaling factors. The optimal set of frequency scaling factors is the set that gives the highest distance according to the objective function (15).

Figures 2a and 2b illustrate the normalized performance and consumed energy for an application running on a homogeneous platform and a heterogeneous platform respectively while increasing the scaling factors. It can be noticed that in a homogeneous platform the search for the optimal scaling factor should start from the maximum frequency because the performance and the consumed energy decrease from the beginning of the plot. On the other hand, in the heterogeneous platform the performance is maintained at the beginning of the plot even if the frequencies of the faster nodes decrease until the computing power of scaled down nodes are lower than the slowest node. In other words, until they reach the higher bound. It can also be noticed that the higher the difference between the faster nodes and the slower nodes is, the bigger the maximum distance between the energy curve and the performance curve is while the scaling factors are varying which results in bigger energy savings. Finally, in a homogeneous platform the energy consumption is increased when the scaling factor is very high. Indeed, the dynamic energy saved by reducing the frequency of the processor is compensated by the significant increase of the execution time and thus the increased of the static energy. On the other hand, in a heterogeneous platform this is not the case.

B. The evaluation of the proposed algorithm

The precision of the proposed algorithm mainly depends on the execution time prediction model defined in (2) and the energy model computed by (11). The energy model is also significantly dependent on the execution time model because the static energy is linearly related to the execution time and the dynamic energy is related to the computation time. So, all the works presented in this paper are based on the execution time model. To verify this model, the predicted execution time was compared to the real execution time over SimGrid/SMPI simulator, v3.10 [24], for all the NAS parallel benchmarks NPB v3.3 [25], running class B on 8 or 9 nodes. The comparison showed that the proposed execution time model is very precise, the maximum normalized difference between the predicted execution time and the real execution time is equal to 0.03 for all the NAS benchmarks.

Since the proposed algorithm is not an exact method, it does not test all the possible solutions (vectors of scaling factors) in the search space. To prove its efficiency, it was compared on small instances to a brute force search algorithm that tests all the possible solutions. The brute force algorithm

Table I: Heterogeneous nodes characteristics

Node type	Simulated GFLOPS	Max Freq. GHz	Min Freq. GHz	Diff. Freq. GHz	Dynamic power	Static power
1	40	2.50	1.20	0.100	20 W	4 W
2	50	2.66	1.60	0.133	25 W	5 W
3	60	2.90	1.20	0.100	30 W	6 W
4	70	3.40	1.60	0.133	35 W	7 W

was applied to different NAS benchmarks classes with different number of nodes. The solutions returned by the brute force algorithm and the proposed algorithm were identical and the proposed algorithm was on average 10 times faster than the brute force algorithm. It has a small execution time: for a heterogeneous cluster composed of four different types of nodes having the characteristics presented in Table I, it takes on average 0.04 ms for 4 nodes and 0.15 ms on average for 144 nodes to compute the best scaling factors vector. The algorithm complexity is $O(F \cdot N)$, where F is the maximum number of available frequencies, and N is the number of computing nodes. The algorithm needs from 12 to 20 iterations to select the best vector of frequency scaling factors that gives the results of the next sections.

VI. Experimental results

To evaluate the efficiency and the overall energy consumption reduction of Algorithm 1, it was applied to the NAS parallel benchmarks NPB v3.3 which is composed of synchronous message passing applications. The experiments were executed on the simulator SimGrid/SMPI which offers easy tools to create a heterogeneous platform and run message passing applications over it. The heterogeneous platform that was used in the experiments, had one core per node because just one process was executed per node. The heterogeneous platform was composed of four types of nodes. Each type of nodes had different characteristics such as the maximum CPU frequency, the number of available frequencies and the computational power, see Table I. The characteristics of these different types of nodes are inspired from the specifications of real Intel processors. The heterogeneous platform had up to 144 nodes and had nodes from the four types in equal proportions, for example if a benchmark was executed on 8 nodes, 2 nodes from each type were used. Since the constructors of CPUs do not specify the dynamic and the static power of their CPUs, for each type of node they were chosen proportionally to its computing power (FLOPS). In the initial heterogeneous platform, while computing with highest frequency, each node consumed an amount of power proportional to its computing power (which corresponds to 80 % of its dynamic power and the remaining 20 % to the static power), the same assumption was made in [5], [21]. Finally, These nodes were connected via an Ethernet network with 1 Gbit/s bandwidth.

A. The experimental results of the scaling algorithm

The proposed algorithm was applied to the seven parallel NAS benchmarks (EP, CG, MG, FT, BT, LU and SP) and the benchmarks were executed with the three classes: A, B and C. However, due to the lack of space in this paper, only the results of the biggest class, C, are presented while being run on different number of nodes, ranging from 8 to 128 or 144 nodes depending on the benchmark being executed. Indeed, the benchmarks CG, MG, LU, EP and FT had to be executed on 1, 2, 4, 8, 16, 32, 64, or 128 nodes. The other benchmarks such as BT and SP had to be executed on 1, 4, 9, 16, 36, 64, or 144 nodes.

The overall energy consumption was computed for each instance according to the energy consumption model (11), with and without applying the algorithm. The execution time was also measured for all these experiments. Then, the energy saving and performance degradation percentages were com-

Table II: Running NAS benchmarks on 8 and 9 nodes

Program name	Execution time/s	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	36.11	3263.49	31.25	7.12	24.13
MG	8.99	953.39	33.78	6.41	27.37
EP	40.39	5652.81	27.04	0.49	26.55
LU	218.79	36149.77	28.23	0.01	28.22
BT	166.89	23207.42	32.32	7.89	24.43
SP	104.73	18414.62	24.73	2.78	21.95
FT	51.10	4913.26	31.02	2.54	28.48

Table III: Running NAS benchmarks on 16 nodes

Program name	Execution time/s	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	31.74	4373.90	26.29	9.57	16.72
MG	5.71	1076.19	32.49	6.05	26.44
EP	20.11	5638.49	26.85	0.56	26.29
LU	144.13	42529.06	28.80	6.56	22.24
BT	97.29	22813.86	34.95	5.80	29.15
SP	66.49	20821.67	22.49	3.82	18.67
FT	37.01	5505.60	31.59	6.48	25.11

Table IV: Running NAS benchmarks on 32 and 36 nodes

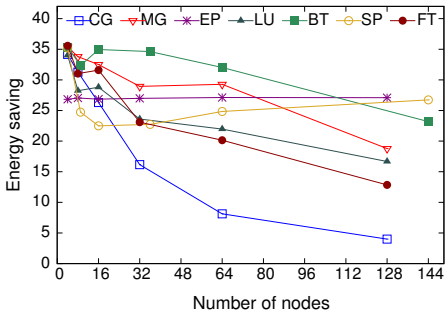
Program name	Execution time/s	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	32.35	6704.21	16.15	5.30	10.85
MG	4.30	1355.58	28.93	8.85	20.08
EP	9.96	5519.68	26.98	0.02	26.96
LU	99.93	67463.43	23.60	2.45	21.15
BT	48.61	23796.97	34.62	5.83	28.79
SP	46.01	27007.43	22.72	3.45	19.27
FT	28.06	7142.69	23.09	2.90	20.19

Table V: Running NAS benchmarks on 64 nodes

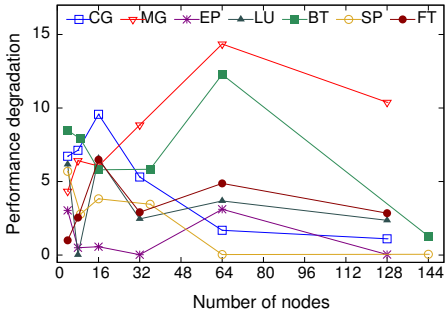
Program name	Execution time/s	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	46.65	17521.83	8.13	1.68	6.45
MG	3.27	1534.70	29.27	14.35	14.92
EP	5.05	5471.11	27.12	3.11	24.01
LU	73.92	101339.16	21.96	3.67	18.29
BT	39.99	27166.71	32.02	12.28	19.74
SP	52.00	49099.28	24.84	0.03	24.81
FT	25.97	10416.82	20.15	4.87	15.28

Table VI: Running NAS benchmarks on 128 and 144 nodes

Program name	Execution time/s	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	56.92	41163.36	4.00	1.10	2.90
MG	3.55	2843.33	18.77	10.38	8.39
EP	2.67	5669.66	27.09	0.03	27.06
LU	51.23	144471.90	16.67	2.36	14.31
BT	37.96	44243.82	23.18	1.28	21.90
SP	64.53	115409.71	26.72	0.05	26.67
FT	25.51	18808.72	12.85	2.84	10.01



(a) Energy saving (%)



(b) Performance degradation (%)

Figure 4: The energy and performance for all NAS benchmarks running with a different number of nodes

puted for each instance. The results are presented in Tables II, III, IV, V and VI. All these results are the average values from many experiments for energy savings and performance degradation. The tables show the experimental results for running the NAS parallel benchmarks on different numbers of nodes. The experiments show that the algorithm significantly reduces the energy consumption (up to 34 %) and tries to limit the performance degradation. They also show that the energy saving percentage decreases when the number of computing nodes increases. This reduction is due to the increase of the communication times compared to the execution times when the benchmarks are run over a higher number of nodes. Indeed, the benchmarks with the same class, C, are executed on different numbers of nodes, so the computation required for each iteration is divided by the number of computing nodes. On the other hand, more communications are required when increasing the number of nodes so the static energy increases linearly according to the communication time and the dynamic power is less relevant in the overall energy consumption. Therefore, reducing the frequency with Algorithm 1 is less effective in reducing the overall energy savings. It can also be noticed that for the benchmarks EP and SP that contain little or no communications, the energy savings are not significantly affected by the high number of nodes. No experiments were conducted using bigger classes than D, because they require a lot of memory (more than 64 GB) when being executed by the simulator on one machine. The maximum distance between the normalized energy curve and the normalized performance for each instance is also shown in the result tables. It decrease in the same way as the energy saving percentage. The tables also show that the performance degradation percentage is not significantly increased when the number of computing nodes is increased because the computation times are small when compared to the communication times.

Figures 4a and 4b present the energy saving and performance degradation respectively for all the benchmarks according to the number of used nodes. As shown in the first plot, the energy saving percentages of the benchmarks MG, LU, BT and FT decrease linearly when the number of nodes increase. While for the EP and SP benchmarks, the energy saving percentage is not affected by the increase of the number of computing nodes, because in these benchmarks there are little or no communications. Finally, the energy saving of the CG benchmark significantly decreases when the number of nodes increase because this benchmark has more communications than the others. The second plot shows that the performance degradation percentages of most of the benchmarks decrease when they run on a big number of nodes because they spend more time communicating than computing, thus, scaling down the frequencies of some nodes has less effect on the performance.

B. *The results for different power consumption scenarios*

The results of the previous section were obtained while using processors that consume during computation an overall power which is 80 % composed of dynamic power and of 20 % of static power. In this section, these ratios are changed and two new power scenarios are considered in order to evaluate how the proposed algorithm adapts itself according to the static and dynamic power values. The two new power scenarios are the following:

- 70 % of dynamic power and 30 % of static power
- 90 % of dynamic power and 10 % of static power

The NAS parallel benchmarks were executed again over processors that follow the new power scenarios. The class C of each benchmark was run over 8 or 9 nodes and the results are presented in Tables VII and VIII. These tables show that the energy saving percentage of the 70 %-30 % scenario is smaller for all benchmarks compared to the energy saving of the 90 %-10 % scenario. Indeed, in the latter more dynamic power is consumed when nodes are running on their maximum frequencies, thus, scaling down the frequency of the nodes results in higher energy savings than in the 70 %-30 % scenario. On the other hand, the performance degradation percentage is smaller in the 70 %-30 % scenario compared to the 90 %-10 % scenario. This is due to the higher static power percentage in the first scenario which makes it more relevant in the overall consumed energy. Indeed, the static energy is related to the execution time and if the performance is degraded the amount of consumed static energy directly increases. Therefore, the proposed algorithm does not really significantly scale down much the frequencies of the nodes in order to limit the increase of the execution time and thus limiting the effect of the consumed static energy.

Both new power scenarios are compared to the old one in Figure 5a. It shows the average of the performance degradation, the energy saving and the distances for all NAS benchmarks of class C running on 8 or 9 nodes. The comparison shows that the energy saving ratio is proportional to the

Table VII: The results of the 70 %-30 % power scenario

Program name	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	4144.21	22.42	7.72	14.70
MG	1133.23	24.50	5.34	19.16
EP	6170.30	16.19	0.02	16.17
LU	39477.28	20.43	0.07	20.36
BT	26169.55	25.34	6.62	18.71
SP	19620.09	19.32	3.66	15.66
FT	6094.07	23.17	0.36	22.81

Table VIII: The results of the 90 %-10 % power scenario

Program name	Energy consumption/J	Energy saving%	Performance degradation%	Distance
CG	2812.38	36.36	6.80	29.56
MG	825.43	38.35	6.41	31.94
EP	5281.62	35.02	2.68	32.34
LU	31611.28	39.15	3.51	35.64
BT	21296.46	36.70	6.60	30.10
SP	15183.42	35.19	11.76	23.43
FT	3856.54	40.80	5.67	35.13

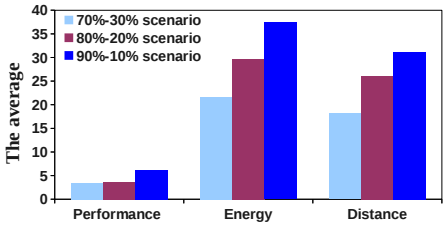
Table IX: Comparing the proposed algorithm

Program name	Energy saving %		Perf. degradation %		Distance	
	EDP	MaxDist	EDP	MaxDist	EDP	MaxDist
CG	27.58	31.25	5.82	7.12	21.76	24.13
MG	29.49	33.78	3.74	6.41	25.75	27.37
LU	19.55	28.33	0.00	0.01	19.55	28.22
EP	28.40	27.04	4.29	0.49	24.11	26.55
BT	27.68	32.32	6.45	7.87	21.23	24.43
SP	20.52	24.73	5.21	2.78	15.31	21.95
FT	27.03	31.02	2.75	2.54	24.28	28.48

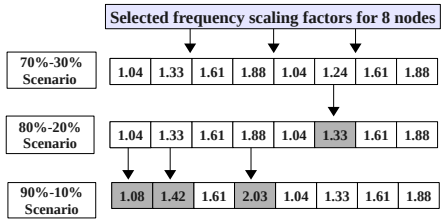
dynamic power ratio: it is increased when applying the 90 %-10 % scenario because at maximum frequency the dynamic energy is the most relevant in the overall consumed energy and can be reduced by lowering the frequency of some processors. On the other hand, the energy saving decreases when the 70 %-30 % scenario is used because the dynamic energy is less relevant in the overall consumed energy and lowering the frequency does not return big energy savings. Moreover, the average of the performance degradation is decreased when using a higher ratio for static power (e.g. 70 %-30 % scenario and 80 %-20 % scenario). Since the proposed algorithm optimizes the energy consumption when using a higher ratio for dynamic power the algorithm selects bigger frequency scaling factors that result in more energy saving but less performance, for example see Figure 5b. The opposite happens when using a higher ratio for static power, the algorithm proportionally selects smaller scaling values which result in less energy saving but also less performance degradation.

C. The comparison of the proposed scaling algorithm

In this section, the scaling factors selection algorithm, called MaxDist, is compared to Spiliopoulos et al. algorithm [7], called EDP. They developed a green governor that regularly applies an online frequency selecting algorithm to reduce the energy consumed by a multicore architecture without degrading much its performance. The algorithm selects the frequencies that minimize the energy and delay products, $EDP = energy \times delay$ using the predicted overall energy consumption and execution time delay for each frequency. To fairly compare both algorithms, the same energy and



(a) Comparison between the results on 8 nodes



(b) Comparison the selected frequency scaling factors of MG benchmark class C running on 8 nodes

Figure 5: The comparison of the three power scenarios

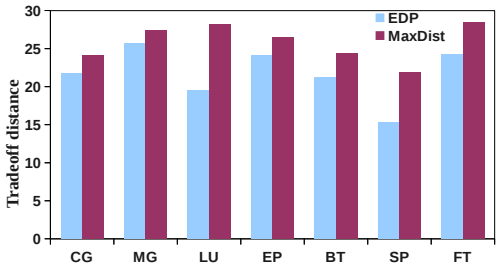


Figure 6: Trade-off comparison for NAS benchmarks class C

execution time models, equations (11) and (7), were used for both algorithms to predict the energy consumption and the execution times. Also Spiliopoulos et al. algorithm was adapted to start the search from the initial frequencies computed using the equation (17). The resulting algorithm is an exhaustive search algorithm that minimizes the EDP and has the initial frequencies values as an upper bound.

Both algorithms were applied to the parallel NAS benchmarks to compare their efficiency. Table IX presents the results of comparing the execution times and the energy consumption for both versions of the NAS benchmarks while running the class C of each benchmark over 8 or 9 heterogeneous nodes. The results show that our algorithm provides better energy savings than Spiliopoulos et al. algorithm, on average it results in 29.76 % energy saving while their algorithm returns just 25.75 %. The average of performance degradation percentage is approximately the same for both algorithms, about 4 %.

For all benchmarks, our algorithm outperforms Spiliopoulos et al. algorithm in terms of energy and performance trade-off, see Figure 6, because it maximizes the distance between the energy saving and the performance degradation values while giving the same weight for both metrics.

VII. Conclusion

In this paper, a new online frequency selecting algorithm has been presented. It selects the best possible vector of frequency scaling factors that gives the maximum distance (optimal trade-off) between the predicted energy and the predicted performance curves for a heterogeneous platform. This algorithm uses a new energy model for measuring and predicting the energy of distributed iterative applications running over heterogeneous platforms. To evaluate the proposed method, it was applied on the NAS parallel benchmarks and executed over a heterogeneous platform simulated by SimGrid. The results of the experiments showed that the algorithm reduces up to 34 % the energy consumption of a message passing iterative method while limiting the degradation of the performance. The algorithm also selects different scaling factors according to the percentage of the computing and communication times, and according to the values of the static and dynamic powers of the CPUs. Finally, the algorithm was compared to Spiliopoulos et al. algorithm and the results showed that it outperforms their algorithm in terms of energy-time trade-off.

In the near future, this method will be applied to real heterogeneous platforms to evaluate its performance in a real study case. It would also be interesting to evaluate its scalability over large scale heterogeneous platforms and measure the energy consumption reduction it can produce. Afterward, we would like to develop a similar method that is adapted to asynchronous iterative applications where each task does not wait for other tasks to finish their works. The development of such a method might require a new energy model because the number of iterations is not known in advance and depends on the global convergence of the iterative system.

Acknowledgment

This work has been partially supported by the Labex ACTION project (contract "ANR-11-LABX-01-01"). Computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté. As a PhD student, Mr. Ahmed Fanfakh, would like to thank the University of Babylon (Iraq) for supporting his work.

References

[1] "TOP500 Supercomputers Sites." [Online]. Available: http://www.top500.org

[2] "U.S. Energy Information Administration, Annual Energy Outlook 2014." [Online]. Available: http://www.eia.gov/

[3] "The Green500 List of Heterogeneous Supercomputing Systems." [Online]. Available: http://www.green500.org

[4] N. B. Rizvandi, J. Taheri, and A. Y. Zomaya, "Some observations on optimal frequency selection in DVFS-based energy consumption minimization," *J. Parallel Distrib. Comput.*, vol. 71, no. 8, pp. 1154–1164, Aug. 2011.

[5] J.-C. Charr, R. Couturier, A. Fanfakh, and A. Giersch, "Dynamic frequency scaling for energy consumption reduction in distributed MPI programs," in *ISPA 2014: 12th IEEE International Symposium on Parallel and Distributed Processing with Applications*. Milan, Italy: IEEE Computer Society, Aug. 2014, pp. 225–230.

[6] H. Shen, J. Lu, and Q. Qiu, "Learning based DVFS for simultaneous temperature, performance and energy management," in *ISQED*, 2012, pp. 747–754.

[7] V. Spiliopoulos, S. Kaxiras, and G. Keramidas, "Green governors: A framework for continuously adaptive dvfs," in *International Green Computing Conference and Workshops (IGCC)*, Jul. 2011, pp. 1–8.

[8] B. Rountree, D. Lowenthal, S. Funk, V. W. Freeh, B. De Supinski, and M. Schulz, "Bounding energy consumption in large-scale MPI programs," in *Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on*, Nov. 2007, pp. 1–9.

[9] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: Adaptive DVFS and thread packing under power caps," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. NY, USA: ACM, 2011, pp. 175–185.

[10] R. Luley, C. Usmail, and M. Barnell, "Energy efficiency evaluation and benchmarking of AFRL's condor high performance computer," DTIC Document, Tech. Rep., 2011.

[11] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, "Greengpu: A holistic approach to energy efficiency in gpu-cpu heterogeneous architectures," in *Parallel Processing (ICPP), 2012 41st International Conference on*, Sep. 2012, pp. 48–57.

[12] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a k20 gpu," in *Parallel Processing (ICPP), 2013 42nd International Conference on*, Oct. 2013, pp. 826–833.

[13] N. Muralimanohar, K. Ramani, and R. Balasubramonian, "Power efficient resource scaling in partitioned architectures through dynamic heterogeneity," in *In Proceedings of ISPASS*, 2006.

[14] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C. zhong Xu, and A. Zomaya, "Energy-aware parallel task scheduling in a cluster," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1661 – 1670, 2013.

[15] K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders, "Blackbox prediction of the impact of DVFS on end-to-end performance of multitier systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 59–63, 2010.

[16] D. Shelepov and A. Fedorova, "Scheduling on heterogeneous multicore processors using architectural signatures," in *Workshop on the Interaction between Operating Systems and Computer Architecture, in conjunction with ISCA*, 2008.

[17] D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[18] J.-J. Chen, K. Huang, and L. Thiele, "Dynamic frequency scaling schemes for heterogeneous clusters under quality of service requirements," *Journal of Information Science and Engineering*, vol. 28, no. 6, pp. 1073–1090, 2012.

[19] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer, "Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01*, ser. IPDPS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 4a–4a.

[20] K. Malkowski, "Co-adapting scientific applications and architectures toward energy-efficient high performance computing," Ph.D. dissertation, The Pennsylvania State University, USA, 2009.

[21] T. Rauber and G. Rünger, "Analytical modeling and simulation of the energy consumption of independent tasks," in *Proceedings of the Winter Simulation Conference*, ser. WSC '12. Winter Simulation Conference, 2012, pp. 245:1–245:13.

[22] J. Zhuo and C. Chakrabarti, "Energy-efficient dynamic task scheduling algorithms for dvs systems," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 2, pp. 17:1–17:25, Jan. 2008.

[23] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003.

[24] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Oct. 2014.

[25] NASA Advanced Supercomputing Division, "NAS parallel benchmarks," Mar. 2012. [Online]. Available: http://www.nas.nasa.gov/publications/npb.html