Optimizing the Energy Consumption of Message Passing Applications with Iterations Executed over Grids

Ahmed Fanfakh, Jean-Claude Charr, Raphaël Couturier, 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: {ahmed.fanfakh_badri_muslim,jean-claude.charr,raphael.couturier,arnaud.giersch}@univ-fcomte.fr

Abstract

In recent years, green computing has become an important topic in the supercomputing research domain. However, the computing platforms are still 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 can be used to reduce the power consumption of the CPU while computing, by lowering its frequency. 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 grids, composed of heterogeneous clusters, 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 application with iterations. The algorithm has a small overhead and works without training or profiling. It uses a new energy model for message passing applications with iterations running on a grid. The proposed algorithm is evaluated on a real grid, the Grid'5000 platform, while running the NAS parallel benchmarks. The experiments on 16 nodes, distributed on three clusters, show that it reduces on average the energy consumption by 30 % while the

performance is on average only degraded by 3.2 %. Finally, the algorithm is compared to an existing method. The comparison results show that it outperforms the latter in terms of energy consumption reduction and performance.
Keywords:
Dynamic voltage and frequency scaling, Grid computing, Green computing and frequency scaling online algorithm.

---

1. 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 June 2015 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 2015 [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 Shoubu-ExaScaler from RIKEN which became the top of the Green500 list in June 2015 [3]. This heterogeneous platform executes more than 7 GFlops per watt while consuming 50.32 kilowatts.

Besides platform improvements, there are many software and hardware techniques to lower the energy consumption of these platforms, such as DVFS, scheduling and other techniques. 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, re-

searchers use different optimization strategies to select the frequency that gives the best trade-off between the energy reduction and performance degradation ratio. In [5] and [6], a frequency selecting algorithm was proposed to reduce the energy consumption of message passing applications with iterations running over homogeneous and heterogeneous clusters respectively. The results of the experiments showed significant energy consumption reductions. All the experimental results were conducted over the SimGrid simulator [7], which offers easy tools to describe homogeneous and heterogeneous platforms, and to simulate the execution of message passing parallel applications over them.

This paper presents the following contributions :

1. two new energy and performance models for message passing synchronous applications with iterations running over a heterogeneous grid platform. Both models take into account communications 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 grid platforms. The algorithm has a very small overhead and does not need any training nor profiling. It uses a new optimization function which simultaneously maximizes the performance and minimizes the energy consumption of a message passing synchronous application with iterations. The algorithm was applied to the NAS parallel benchmarks and evaluated over a real testbed, the Grid'5000 platform [8].

This paper is organized as follows: Section 2 presents some related works from other authors. Section 3 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 grid platform. Section 4 presents the energy-performance objective function that maximizes the reduction of energy consumption while minimizing the degradation of the program's performance. Section 5 details the proposed frequencies selecting algorithm. Section 6 presents the results of applying the algorithm on the NAS

parallel benchmarks and executing them on the Grid'5000 testbed. It also evaluates the algorithm over multi-core per node architectures and over three different power scenarios. Moreover, it shows the comparison results between the proposed method and an existing method. Finally, in Section 7 the paper ends with a summary and some future works.

2. 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 [9, 10]. Others used offline methods that may need to run the application and profile it before selecting the new frequency, such as [11, 12]. 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 the energy consumption of message passing synchronous applications with iterations running over heterogeneous grid 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. [13], 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 [14], Kai Ma et al. developed a scheduling algorithm that distributes workload 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 [15], 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. [16] 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. [17] 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 [18], 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 [19] and [20], the best frequencies for a specified heterogeneous cluster are selected offline using some heuristic. Chen et al. [21] used a greedy dynamic programming approach to minimize the power consumption of heterogeneous servers while respecting given time constraints. This approach had considerable overhead.

3. The performance and energy consumption measurements on heterogeneous grid architecture

3.1. The execution time of message passing distributed applications with iterations on a heterogeneous platform

In this paper, we are interested in reducing the energy consumption of message passing distributed synchronous applications with iterations running over heterogeneous grid platforms. A heterogeneous grid platform could be defined as a collection of heterogeneous computing clusters interconnected via a long distance network which has lower bandwidth and higher latency than the local networks of the clusters. Each computing cluster in the grid is composed of homogeneous nodes that are connected together via high speed network. Therefore, each cluster has different characteristics such as computing power (FLOPS), energy consumption, CPU's frequency range, network bandwidth and latency.

The overall execution time of a distributed synchronous application with iterations running over a heterogeneous grid consists of the sum of the computation time and the communication time for every iteration on a node. However, nodes from distinct clusters in a grid have different computing powers, thus while the application, 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. Therefore, the overall execution time of the program is the execution time of the slowest task which has the highest computation time and almost no slack time. For example, in Figure 1, task 1 is the slower task and it does not have to wait for the other nodes to communicate with them because they all finish their computations before it.

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

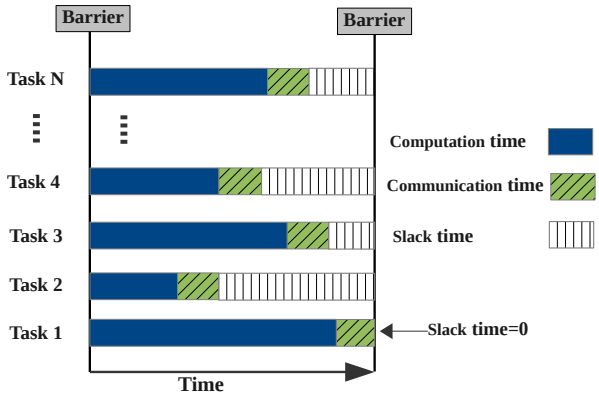


Figure 1: Parallel tasks on a heterogeneous platform

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).

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

where F_{max} is the maximum frequency before applying any DVFS and F_{new} is the new frequency after applying DVFS.

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 [22]. 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 grid each cluster has different characteristics, especially different frequency gears, when applying DVFS operations on the nodes of these clusters, they may get different scaling factors represented by a scaling vector: $(S_{11}, S_{12}, \ldots, S_{NM_i})$ where S_{ij} is the scaling factor of processor j in clus-

ter i . To be able to predict the execution time of message passing synchronous applications with iterations running over a heterogeneous grid, 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 Equation (2).

$$T_{New} = \max_{\substack{i=1,...N \\ j=1,...,M_i}} (T_{cpOld_{ij}} \cdot S_{ij}) + \min_{j=1,...,M_h} (T_{cm_{hj}}) \quad (2)$$

where N is the number of clusters in the grid, M_i is the number of nodes in cluster i, $T_{cpOld_{ij}}$ is the computation time of processor j in the cluster i and $T_{cm_{hj}}$ is the communication time of processor j in the cluster h during the first iteration. The execution time for one iteration is equal to the sum of the maximum computation time for all nodes with the new scaling factors and the communication time of the slowest node without slack time during one iteration. The slowest node in cluster h is the node which takes the maximum execution time to execute an iteration before scaling down its frequency. It means that only the communication time without any slack time is taken into account. Therefore, the execution time of the application is equal to the execution time of one iteration as in Equation (2) multiplied by the number of iterations of that application.

This model is an adaptation of the one developed in [5] which predicts the execution time of message passing applications with iterations running on homogeneous clusters. In a homogeneous cluster only one scaling factor denoted as S was used because all the nodes in the cluster have the same computing power. In a heterogeneous cluster, each node may have a different scaling factor denoted as (S_i) where i is the index of the node. In a grid, each node in each cluster may have a scaling factor. The whole set of scaling factors of all the computing nodes in the grid is denoted by a two dimensional array of scales $(S_{11}, S_{12}, \ldots, S_{NM_i})$ where N is the number of used clusters and M_i is the

number of nodes in cluster i.

The execution time model, Equation 2, is used in the algorithm presented in section 5. The latter selects the scaling factors that optimize both the energy consumption and the performance of message passing applications with iterations running on grids.

3.2. Energy model for heterogeneous grid platform

Many researchers [23, 24, 25, 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 (3).

$$P_d = \alpha \cdot C_L \cdot V^2 \cdot F \tag{3}$$

The static power P_s captures the leakage power as follows:

$$P_s = V \cdot N_{trans} \cdot K_{design} \cdot I_{leak} \tag{4}$$

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:

$$E_{ind} = P_d \cdot T_{cp} + P_s \cdot T \tag{5}$$

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 [26]. 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 [24]. 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:

$$F_{new} = S^{-1} \cdot F_{max} \tag{6}$$

Replacing F_{new} in (3) as in (6) gives the following equation for dynamic power consumption:

$$\begin{aligned} P_{dNew} = \alpha \cdot C_L \cdot V^2 \cdot F_{new} = \alpha \cdot C_L \cdot \beta^2 \cdot {F_{new}}^3 \\ = \alpha \cdot C_L \cdot V^2 \cdot F_{max} \cdot S^{-3} = P_{dOld} \cdot S^{-3} \end{aligned} \tag{7}$$

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

According to (7) the dynamic power is reduced by a factor of S^{-3} when reducing the frequency by a factor of S [24]. 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:

$$E_{dNew} = P_{dOld} \cdot S^{-3} \cdot (T_{cp} \cdot S) = S^{-2} \cdot P_{dOld} \cdot T_{cp} \tag{8}$$

The static power is related to the power leakage of the CPU and is consumed during computation and even when idle. As in [24, 25], 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:

$$E_S = P_s \cdot (T_{cp} \cdot S + T_{cm}) \tag{9}$$

In the considered heterogeneous grid platform, each node j in cluster i may have different dynamic and static powers from the nodes of the other clusters, noted as $P_{d_{ij}}$ and $P_{s_{ij}}$ respectively. Moreover, even if the distributed message passing application with iterations is load balanced, the computation time of each CPU j in cluster i noted $T_{cp_{ij}}$ may be slightly different due to the delay caused by the scheduler of the operating system. Therefore, different frequency scaling factors may be computed in order to decrease the overall energy consumption of the application and reduce the slack times. The communication time of a processor j in cluster i is noted as $T_{cm_{ij}}$ 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 (8), the static energy is computed as the sum of the execution time of one iteration multiplied by the static power of each processor. The CPU during the communication times consumes only the static power. While in the computation times, it consumes both the dynamic and the static powers, for more information refer to [22]. The overall energy consumption of a message passing distributed application executed over a heterogeneous grid platform during one iteration is the summation of all dynamic and static energies for M_i processors in N clusters. It is computed as follows:

$$E = \sum_{i=1}^{N}\sum_{i=1}^{M_i}(S_{ij}^{-2} \cdot P_{d_{ij}} \cdot T_{cp_{ij}}) + \sum_{i=1}^{N}\sum_{j=1}^{M_i}(P_{s_{ij}} \cdot \left(\max_{\substack{i=1,\dots N \\ j=1,\dots,M_i}}(T_{cp_{ij}} \cdot S_{ij}) + \min_{j=1,\dots M_h}(T_{cm_{hj}})\right)) \tag{10}$$

Reducing the frequencies of the processors according to the vector of scaling

factors $(S_{11}, S_{12}, \ldots, S_{NM_i})$ may degrade the performance of the application and thus, increase the static energy because the execution time is increased [27]. The overall energy consumption for a synchronous application with iterations can be measured by measuring the energy consumption for one iteration as in (10) multiplied by the number of iterations of that application.

4. 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 works, [5] and [6], two methods that select the optimal frequency scaling factors for a homogeneous and a heterogeneous cluster respectively, were proposed. Both methods selects the frequencies that gives the best trade-off between energy consumption reduction and performance for message passing synchronous applications with iterations. In this work we are interested in grids that are composed of heterogeneous clusters. The nodes from distinct clusters may have different characteristics such as dynamic power, static power, computation power, frequencies range, network latency and bandwidth. 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 [22]. 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:

$$P_{Norm} = \frac{T_{New}}{T_{Old}} \tag{11}$$

where $Tnew$ is computed as in (2) and $Told$ is computed as in (12).

$$T_{Old} = \max_{\substack{i=1,\dots N \\ j=1,\dots,M_i}} (T_{cpOld_{ij}}) + \min_{j=1,\dots,M_h} (T_{cm_{hj}}) \tag{12}$$

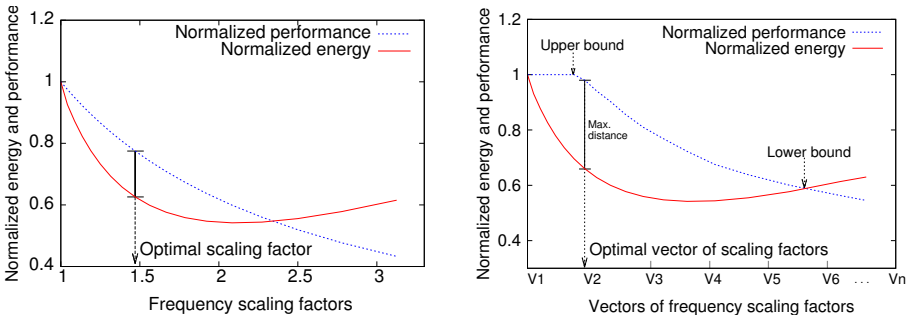
In the same way, the energy is normalized by computing the ratio between the consumed energy while scaling down the frequency and the consumed energy with maximum frequency for all nodes:

$$E_{Norm} = \frac{E_{Reduced}}{E_{Original}} \tag{13}$$

where $E_{Reduced}$ is computed using (10) and $E_{Original}$ is computed as in (14).

$$E_{Original} = \sum_{i=1}^{N} \sum_{j=1}^{M_i} (P_{d_{ij}} \cdot T_{cp_{ij}}) + \sum_{i=1}^{N} \sum_{j=1}^{M_i} (P_{s_{ij}} \cdot T_{Old}) \tag{14}$$

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 (11) and (13), the vector of frequency scaling factors $S_{11}, S_{12}, \dots, S_{NM_i}$ reduces both the energy and the execution time, but the main objective is to produce maximum energy reduction with minimum execution time reduction.



(a) Homogeneous cluster (b) Heterogeneous grid

Figure 2: The energy and performance relation

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 $(S_{11}, S_{12}, \ldots, S_{NM_i})$. Therefore, the equation of the normalized execution time is inverted which gives the normalized performance equation, as follows:

$$P_{Norm} = \frac{T_{Old}}{T_{New}} \tag{15}$$

Then, the objective function can be modeled in order to find the maximum distance between the energy curve (13) and the performance curve (15) 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 and Figure 2b. Then the objective function has the following form:

$$MaxDist = \max_{\substack{i=1,\ldots N \\ j=1,\ldots,M_i \\ k=1,\ldots,F_j}} (\overbrace{P_{Norm}(S_{ijk})}^{\text{Maximize}} - \overbrace{E_{Norm}(S_{ijk})}^{\text{Minimize}}) \tag{16}$$

where N is the number of clusters, M_i is the number of nodes in the cluster i and F_j is the number of available frequencies in the node j. Then, the optimal set of scaling factors that satisfies (16) 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

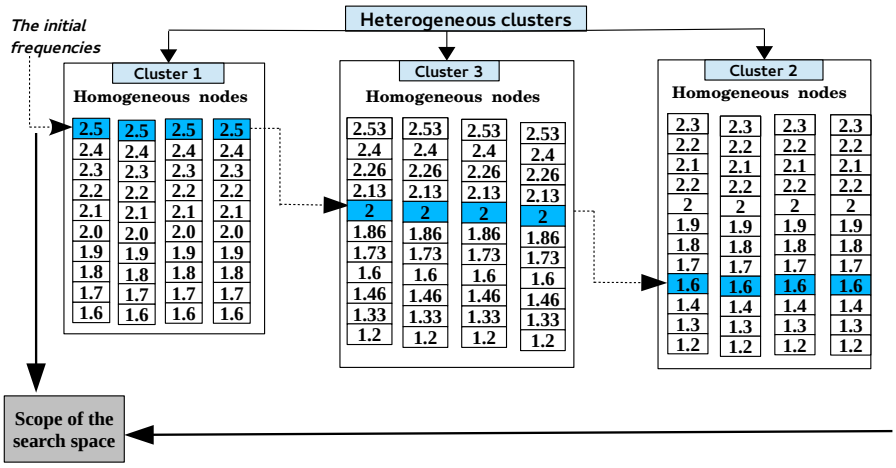


Figure 3: Selecting the initial frequencies in a grid platform

achieved when the energy curve has a convex form as shown in [25, 24, 9].

5. The scaling factors selection algorithm for grids

In this section, the scaling factors selection algorithm for grids, Algorithm 1, is presented. It selects the vector of frequency scaling factors that gives the best trade-off between minimizing the energy consumption and maximizing the performance of a message passing synchronous application with iterations executed on a grid. It works online during the execution time of the application. It uses information gathered during the first iteration such as the computation time and the communication 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 (16). 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 application.

The algorithm takes into account this problem and tries to reduce these slack times when selecting the vector of the frequency scaling factors. At first,

Algorithm 1 Scaling factors selection algorithm

Require:

N number of clusters in the grid.

M_i number of nodes in each cluster.

$T_{cp_{ij}}$ array of all computation times for all nodes during one iteration and with the highest frequency.

$T_{cm_{ij}}$ array of all communication times for all nodes during one iteration and with the highest frequency.

$F_{max_{ij}}$ array of the maximum frequencies for all nodes.

$P_{d_{ij}}$ array of the dynamic powers for all nodes.

$P_{s_{ij}}$ array of the static powers for all nodes.

$F_{diff_{ij}}$ array of the differences between two successive frequencies for all nodes.

Ensure: $S_{opt_{11}}, S_{opt_{12}} \ldots, S_{opt_{NM_i}}$, a vector of scaling factors that gives the optimal trade-off between energy consumption and execution time

1: $S_{cp_{ij}} \leftarrow \frac{\max_{i=1,2,\ldots,N}(\max_{j=1,2,\ldots,M_i}(T_{cp_{ij}}))}{T_{cp_{ij}}}$
2: $F_{ij} \leftarrow \frac{F_{max_{ij}}}{S_{cp_i}},\ i = 1, 2, \cdots, N,\ j = 1, 2, \ldots, M_i.$
3: Round the computed initial frequencies F_i to the closest available frequency for each node.
4: **if** (not the first frequency) **then**
5: $F_{ij} \leftarrow F_{ij} + F_{diff_{ij}},\ i = 1, \ldots, N,\ j = 1, \ldots, M_i.$
6: **end if**
7: $T_{Old} \leftarrow$ computed as in Equation 12.
8: $E_{Original} \leftarrow$ computed as in Equation 14.
9: $S_{opt_{ij}} \leftarrow 1,\ i = 1, \ldots, N,\ j = 1, \ldots, M_i.$
10: $Dist \leftarrow 0$
11: **while** (all nodes have not reached their minimum frequency **or** $P_{Norm} - E_{Norm} < 0$) **do**
12: **if** (not the last freq. **and** not the slowest node) **then**
13: $F_{ij} \leftarrow F_{ij} - F_{diff_{ij}},\ i = 1, \ldots, N,\ j = 1, \ldots, M_i.$
14: $S_{ij} \leftarrow \frac{F_{max_{ij}}}{F_{ij}},\ i = 1, \ldots, N,\ j = 1, \ldots, M_i.$
15: **end if**
16: $T_{New} \leftarrow$ computed as in Equation 2.
17: $E_{Reduced} \leftarrow$ computed as in Equation 10.
18: $P_{Norm} \leftarrow \frac{T_{Old}}{T_{New}},\ E_{Norm} \leftarrow \frac{E_{Reduced}}{E_{Original}}$
19: **if** $(P_{Norm} - E_{Norm} > Dist)$ **then**
20: $S_{opt_{ij}} \leftarrow S_{ij},\ i = 1, \ldots, N,\ j = 1, \ldots, M_i.$
21: $Dist \leftarrow P_{Norm} - E_{Norm}$
22: **end if**
23: **end while**
24: Return $S_{opt_{11}}, S_{opt_{12}}, \ldots, S_{opt_{NM_i}}$

Algorithm 2 DVFS algorithm

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

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:

$$S_{cp_{ij}} = \frac{\max\limits_{\substack{i=1,\ldots N \\ j=1,\ldots,M_i}} (T_{cp_{ij}})}{T_{cp_{ij}}} \tag{17}$$

Using the initial frequency scaling factors computed in (17), the algorithm computes the initial frequencies for all nodes as a ratio between the maximum frequency of node and its computed scaling factor as follows:

$$F_{ij} = \frac{F_{max_{ij}}}{S_{cp_{ij}}},\ i = 1, 2, \ldots, N,\ j = 1, \ldots, M_i \tag{18}$$

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 powers 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 higher frequencies 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 until reaching the nodes' minimum frequencies or lower bounds. A node's frequency is considered its lower bound if the computed distance between the energy and performance at this frequency is less than zero. A negative distance means that the performance degradation ratio is higher than the energy saving ratio. In this situation, the algorithm must stop the downward search because it has reached the lower bound and it is useless to test the lower frequencies. Indeed, they will all give worse distances.

Therefore, the algorithm iterates on all remaining frequencies, from the higher bound until all nodes reach their minimum frequencies or their lower bounds, to compute the overall energy consumption and performance and selects the optimal vector of the frequency scaling factors. At each iteration the algorithm determines the slowest node according to 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 16.

Figures 2a and 2b illustrate the normalized performance and consumed energy for an application running on a homogeneous cluster and a grid platform respectively while increasing the scaling factors. It can be noticed that in a homogeneous cluster 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 grid 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. 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, which results in bigger energy savings.

6. Experimental results

While in [6] the energy model and the scaling factors selection algorithm were applied to a heterogeneous cluster and evaluated over the SimGrid simulator [7], in this paper real experiments were conducted over the Grid'5000 platform.

6.1. Grid'5000 architecture and power consumption

Grid'5000 [8] is a large-scale testbed that consists of ten sites distributed all over metropolitan France and Luxembourg. All the sites are connected together via a special long distance network called RENATER, which is the French National Telecommunication Network for Technology. Each site of the grid is composed of a few heterogeneous computing clusters and each cluster contains many homogeneous nodes. In total, Grid'5000 has about one thousand heterogeneous nodes and eight thousand cores. In each site, the clusters and their nodes are connected via high speed local area networks. Two types of local networks are used, Ethernet or Infiniband networks which have different characteristics in terms of bandwidth and latency.

Since Grid'5000 is dedicated to testing, contrary to production grids it allows a user to deploy its own customized operating system on all the booked nodes. The user could have root rights and thus apply DVFS operations while executing a distributed application. Moreover, the Grid'5000 testbed provides at some sites a power measurement tool to capture the power consumption for each node in those sites. The measured power is the overall consumed power by all the components of a node at a given instant. For more details refer to [28]. In order to correctly measure the CPU power of one core in a node j, firstly, the power consumed by the node while being idle at instant y, noted as $P_{idle_{jy}}$, was measured. Then, the power was measured while running a single thread benchmark with no communication (no idle time) over the same node with its

CPU scaled to the maximum available frequency. The latter power measured at time x with maximum frequency for one core of node j is noted $P_{max_{jx}}$. The difference between the two measured power consumptions represents the dynamic power consumption of that core with the maximum frequency, see Figure 5.

The dynamic power P_{d_j} is computed as in Equation 19

$$P_{d_j} = \max_{x=\beta_1,\ldots\beta_2}(P_{max_{jx}}) - \min_{y=\Theta_1,\ldots\Theta_2}(P_{idle_{jy}}) \tag{19}$$

where P_{d_j} is the dynamic power consumption for one core of node j, $\{\beta_1, \beta_2\}$ is the time interval for the measured maximum power values, $\{\Theta_1, \Theta_2\}$ is the time interval for the measured idle power values. Therefore, the dynamic power of one core is computed as the difference between the maximum measured value in maximum powers vector and the minimum measured value in the idle powers vector.

On the other hand, the static power consumption by one core is a part of the measured idle power consumption of the node. Since in Grid'5000 there is no way to measure precisely the consumed static power and in [5, 6, 24] it was assumed that the static power represents a ratio of the dynamic power, the value of the static power is assumed as 20% of dynamic power consumption of the core.

In the experiments presented in the following sections, two sites of Grid'5000 were used, Lyon and Nancy sites. These two sites have in total seven different clusters as shown on Figure 4.

Four clusters from the two sites were selected in the experiments: one cluster from Lyon's site, Taurus, and three clusters from Nancy's site, Graphene, Griffon and Graphite. Each one of these clusters has homogeneous nodes inside, while nodes from different clusters are heterogeneous in many aspects such as: computing power, power consumption, available frequency ranges and local network features: the bandwidth and the latency. Table 1 shows the detailed characteristics of these four clusters. Moreover, the dynamic powers were computed

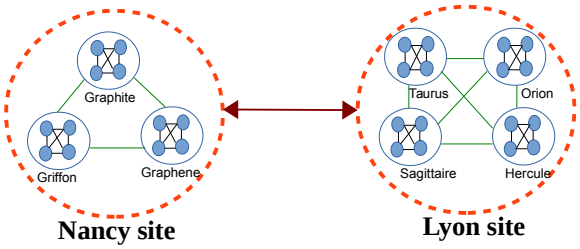


Figure 4: The selected two sites of Grid'5000

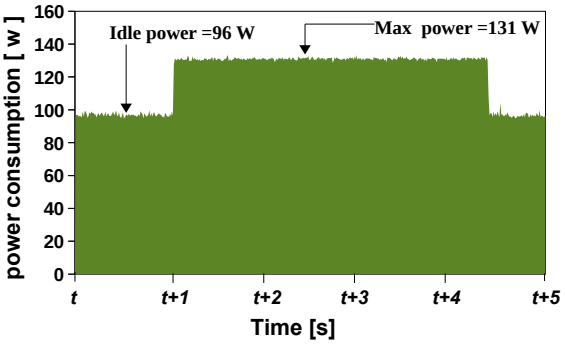


Figure 5: The power consumption by one core from the Taurus cluster

using Equation 19 for all the nodes in the selected clusters and are presented in Table 1.

The energy model and the scaling factors selection algorithm were applied to the NAS parallel benchmarks v3.3 [29] and evaluated over Grid'5000. The benchmark suite contains seven applications: CG, MG, EP, LU, BT, SP and FT. These benchmarks are considered as message passing applications with iterations because the same block of operations is executed many times. These applications have different computations and communications ratios and strategies which make them good testbed applications to evaluate the proposed algorithm and energy model. The benchmarks have seven different classes, S, W, A, B, C, D and E, that represent the size of the problem that the method solves. In the next sections, the class D was used for all the benchmarks in all the experiments.

Table 1: The characteristics of the CPUs in the selected clusters

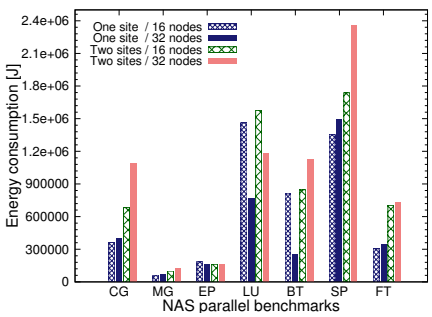
Cluster Name	CPU model	Max Freq. GHz	Min Freq. GHz	Diff. Freq. GHz	Cores per CPU	Dynamic power of one core
Taurus	Intel Xeon E5-2630	2.3	1.2	0.1	6	35 W
Graphene	Intel Xeon X3440	2.53	1.2	0.133	4	23 W
Griffon	Intel Xeon L5420	2.5	2	0.5	4	46 W
Graphite	Intel Xeon E5-2650	2	1.2	0.1	8	35 W

6.2. The experimental results of the scaling algorithm

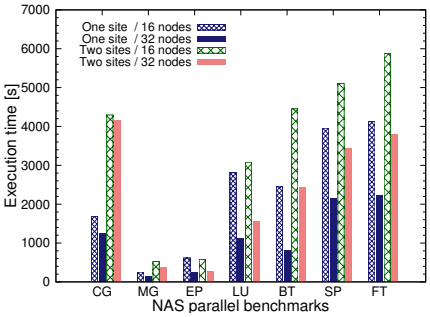
In this section, the results of the application of the scaling factors selection algorithm 1 to the NAS parallel benchmarks are presented. Each experiment has been executed many times and the results presented in the figures are the average values of many executions. As mentioned previously, the experiments were conducted over two sites of Grid'5000, Lyon and Nancy sites. Two scenarios were considered while selecting the clusters from these two sites :

- In the first scenario, nodes from two sites and three heterogeneous clusters were selected. The two sites are connected via a long distance network.
- In the second scenario nodes from three clusters located in one site, Nancy site, were selected.

The main reason for using these two scenarios is to evaluate the influence of long distance communications (higher latency) on the performance of the scaling factors selection algorithm. Indeed, in the first scenario the computations to communications ratio is very low due to the higher communication times which reduces the effect of the DVFS operations.



(a) The energy consumption by the nodes wile executing the NAS benchmarks over different scenarios



(b) The execution times of the NAS benchmarks over different scenarios

Figure 6: The energy consumption and execution time of NAS Benchmarks over different scenarios

The NAS parallel benchmarks are executed over 16 and 32 nodes for each scenario. The number of participating computing nodes from each cluster is different because all the selected clusters do not have the same available number of nodes and all benchmarks do not require the same number of computing nodes. Table 2 shows the number of nodes used from each cluster for each scenario.

Table 2: The different grid scenarios

Scenario name	The participating clusters		
	Cluster	Site	Nodes per cluster
Two sites / 16 nodes	Taurus	Lyon	5
	Graphene	Nancy	5
	Griffon	Nancy	6
Two sites / 32 nodes	Taurus	Lyon	10
	Graphene	Nancy	10
	Griffon	Nancy	12
One site / 16 nodes	Graphite	Nancy	4
	Graphene	Nancy	6
	Griffon	Nancy	6
One site / 32 nodes	Graphite	Nancy	4
	Graphene	Nancy	14
	Griffon	Nancy	14

The NAS parallel benchmarks are executed over these two platforms with different number of nodes, as in Table 2. The overall energy consumption of all the benchmarks solving the class D instance and using the proposed frequency selection algorithm is measured using the equation of the reduced energy consumption, Equation 10. This model uses the measured dynamic power showed in Table 1 and the static power is assumed to be equal to 20% of the dynamic power as in [24]. The execution time is measured for all the benchmarks over these different scenarios.

The energy consumptions and the execution times for all the benchmarks are presented in Figures 6a and 6b respectively.

For the majority of the benchmarks, the energy consumed while executing the NAS benchmarks over one site scenario for 16 and 32 nodes is lower than the energy consumed while using two sites. The long distance communications between the two distributed sites increase the idle time, which leads to more static energy consumption.

The execution times of these benchmarks over one site with 16 and 32 nodes are also lower than those of the two sites scenario. Moreover, most of the benchmarks running over the one site scenario have their execution times approximately halved when the number of computing nodes is doubled from 16 to 32 nodes (linear speed up according to the number of the nodes).

However, the execution times and the energy consumptions of the EP and MG benchmarks, which have no or small communications, are not significantly affected in both scenarios, even when the number of nodes is doubled. On the other hand, the communication times of the rest of the benchmarks increase when using long distance communications between two sites or when increasing the number of computing nodes.

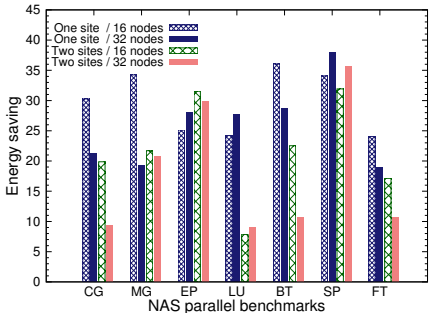
The energy saving percentage is computed as the ratio between the reduced energy consumption, Equation 10, and the original energy consumption, Equation 14, for all benchmarks as in Figure 7a. This figure shows that the energy saving percentages of one site scenario for 16 and 32 nodes are bigger than those of the two sites scenario which is due to the higher computations to com-

munications ratio in the first scenario than in the second one. Moreover, the frequency selecting algorithm selects smaller frequencies when the computation times are bigger than the communication times which results in a lower energy consumption. Indeed, the dynamic consumed power is exponentially related to the CPU's frequency value. On the other hand, the increase in the number of computing nodes can increase the communication times and thus produces less energy saving depending on the benchmarks being executed. The results of benchmarks CG, MG, BT and FT show more energy saving percentage in the one site scenario when executed over 16 nodes than over 32 nodes. LU and SP consume more energy with 16 nodes than 32 nodes on one site because their computations to communications ratio is not affected by the increase of the number of local communications.

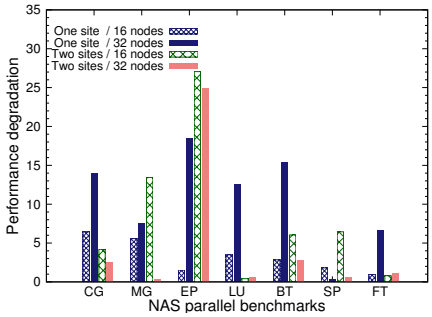
The energy saving percentage is reduced for all the benchmarks because of the long distance communications in the two sites scenario, except for the EP benchmark which has no communication. Therefore, the energy saving percentage of this benchmark is dependent on the maximum difference between the computing powers of the heterogeneous computing nodes, for example in the one site scenario, the graphite cluster is selected but in the two sites scenario this cluster is replaced with the Taurus cluster which is more powerful. Therefore, the energy savings of the EP benchmark are bigger in the two sites scenario due to the higher maximum difference between the computing powers of the nodes.

In fact, high differences between the nodes' computing powers make the proposed frequencies selecting algorithm select smaller frequencies for the powerful nodes which produces less energy consumption and thus more energy saving. The best energy saving percentage was obtained in the one site scenario with 16 nodes, the energy consumption was on average reduced up to 30%.

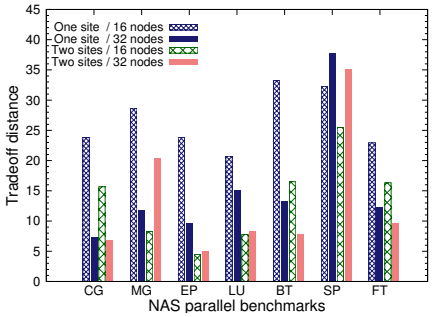
Figure 7b presents the performance degradation percentages for all benchmarks over the two scenarios. The performance degradation percentage for the benchmarks running on two sites with 16 or 32 nodes is on average equal to 8.3% or 4.7% respectively. For this scenario, the proposed scaling algorithm selects smaller frequencies for the executions with 32 nodes without significantly



(a) The energy reduction while executing the NAS benchmarks over different scenarios



(b) The performance degradation of the NAS benchmarks over different scenarios



(c) The trade-off distance between the energy reduction and the performance of the NAS benchmarks over different scenarios

Figure 7: The experimental results of different scenarios

degrading their performance because the communication times are high with 32 nodes which results in smaller computations to communications ratio. On the other hand, the performance degradation percentage for the benchmarks running on one site with 16 or 32 nodes is on average equal to 3.2% and 10.6% respectively. In contrary to the two sites scenario, when the number of computing nodes is increased in the one site scenario, the performance degradation percentage is increased. Therefore, doubling the number of computing nodes when the communications occur in high speed network does not decrease the computations to communication ratio.

The performance degradation percentage of the EP benchmark after applying the scaling factors selection algorithm is the highest in comparison to the

other benchmarks. Indeed, in the EP benchmark, there are no communication and no slack times and its performance degradation percentage only depends on the frequencies values selected by the algorithm for the computing nodes. The rest of the benchmarks showed different performance degradation percentages which decrease when the communication times increase and vice versa.

Figure 7c presents the distance percentage between the energy saving and the performance degradation for each benchmark over both scenarios. The trade-off distance percentage can be computed as in Equation 16. The one site scenario with 16 nodes gives the best energy and performance trade-off, on average it is equal to 26.8%. The one site scenario using both 16 and 32 nodes had better energy and performance trade-off comparing to the two sites scenario because the former has high speed local communications which increase the computations to communications ratio and the latter uses long distance communications which decrease this ratio.

Finally, the best energy and performance trade-off depends on all of the following: 1) the computations to communications ratio when there are communications and slack times, 2) the heterogeneity of the computing powers of the nodes and 3) the heterogeneity of the consumed static and dynamic powers of the nodes.

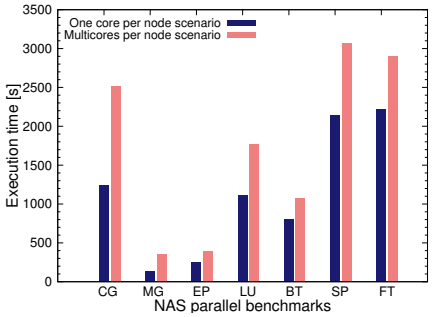
6.3. The experimental results over multi-core clusters

The clusters of Grid'5000 have different number of cores embedded in their nodes as shown in Table 1. In this section, the proposed scaling algorithm is evaluated over the Grid'5000 platform while using multi-cores nodes selected according to the one site scenario described in Section 6.2. The one site scenario uses 32 cores from multi-core nodes instead of 32 distinct nodes. For example if the participating number of cores from a certain cluster is equal to 14, in the multi-core scenario 4 nodes are selected and 3 or 4 cores from each node are used. The platforms with one core per node and multi-core nodes are shown in Table 3. The energy consumptions and execution times of running the class D of the NAS parallel benchmarks over these two different platforms are presented

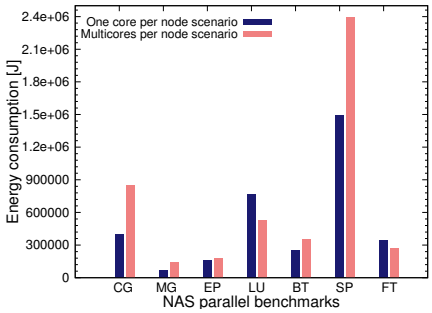
in Figures 8b and 8a respectively.

Table 3: The multi-core scenarios

Scenario name	Cluster name	Nodes per cluster	Cores per node
One core per node	Graphite	4	1
	Graphene	14	1
	Griffon	14	1
Multi-core per node	Graphite	1	4
	Graphene	4	3 or 4
	Griffon	4	3 or 4



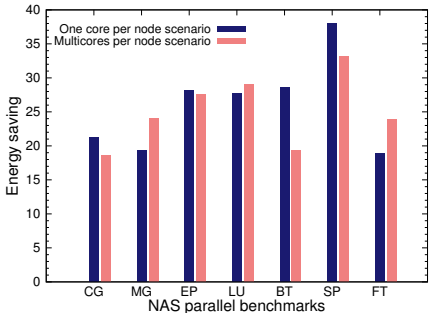
(a) Comparing the execution times of running the NAS benchmarks over one core and multi-core scenarios



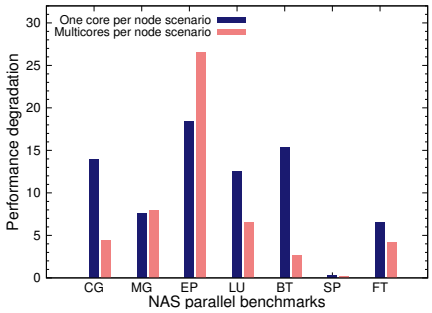
(b) Comparing the energy consumptions of running the NAS benchmarks over one core and multi-core scenarios

Figure 8: The energy consumptions and execution times of the NAS benchmarks running over one core and multi-core per node architectures

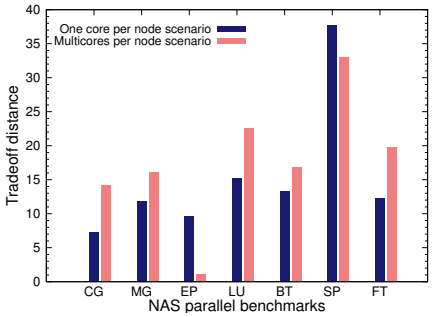
The execution times for most of the NAS benchmarks are higher over the multi-core per node scenario than over the single core per node scenario. Indeed, the communication times are higher in the multi-core scenario than in the latter scenario because all the cores of a node share the same node network link which can be saturated when running communication bound applications. Moreover, the cores of a node share the memory bus which can be also saturated and become a bottleneck. Moreover, the energy consumptions of the NAS benchmarks are lower over the one core scenario than over the multi-core scenario because the first scenario had less execution time than the latter which results in less static energy being consumed. The computations to communications ratios of the NAS benchmarks are higher over the one site one core scenario when



(a) The energy saving of running NAS benchmarks over one core and multicore scenarios



(b) The performance degradation of running NAS benchmarks over one core and multicore scenarios



(c) The trade-off distance of running NAS benchmarks over one core and multicore scenarios

Figure 9: The experimental results of one core and multi-core scenarios

compared to the ratio of the multi-core scenario. More energy reduction can be gained when this ratio is big because it pushes the proposed scaling algorithm to select smaller frequencies that decrease the dynamic power consumption. These experiments also showed that the energy consumption and the execution times of the EP and MG benchmarks do not change significantly over these two scenarios because there are no or small communications. Contrary to EP and MG, the energy consumptions and the execution times of the rest of the benchmarks vary according to the communication times that are different from one scenario to the other.

The energy saving percentages of all the NAS benchmarks running over these

two scenarios are presented in Figure 9a. The figure shows that the energy saving percentages in the one core and the multi-core scenarios are approximately equivalent, on average they are equal to 25.9% and 25.1% respectively. The energy consumption is reduced at the same rate in the two scenarios when compared to the energy consumption of the executions without DVFS.

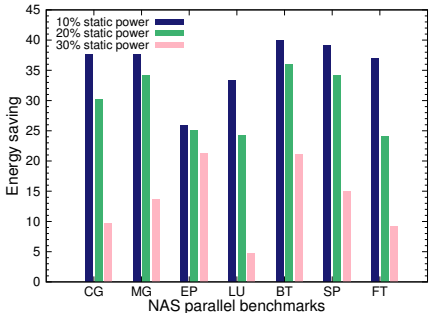
The performance degradation percentages of the NAS benchmarks are presented in Figure 9b. It shows that the performance degradation percentages are higher for the NAS benchmarks executed over the one core per node scenario (on average equal to 10.6%) than over the multi-core scenario (on average equal to 7.5%). The performance degradation percentages over the multi-core scenario are lower because the computations to communications ratios are smaller than the ratios of the other scenario.

The trade-off distances percentages of the NAS benchmarks over both scenarios are presented in Figure 9c. These trade-off distances between energy consumption reduction and performance are used to verify which scenario is the best in both terms at the same time. The figure shows that the trade-off distance percentages are on average bigger over the multi-core scenario (17.6%) than over the one core per node scenario (15.3%).

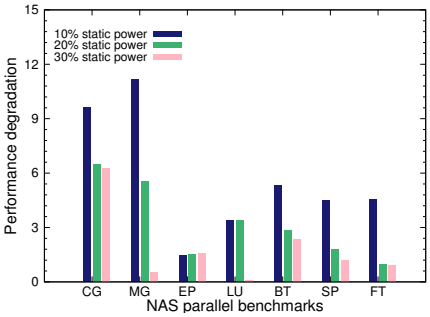
6.4. Experiments with different static power scenarios

In Section 6.1, since it was not possible to measure the static power consumed by a CPU, the static power was assumed to be equal to 20% of the measured dynamic power. This power is consumed during the whole execution time, during computation and communication times. Therefore, when the DVFS operations are applied by the scaling algorithm and the CPUs' frequencies lowered, the execution time might increase and consequently the consumed static energy will be increased too.

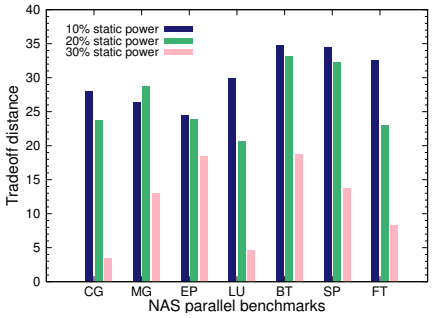
The aim of this section is to evaluate the scaling algorithm while assuming different values of static powers. In addition to the previously used percentage of static power, two new static power ratios, 10% and 30% of the measured dynamic power of the core, are used in this section. The experiments have



(a) The energy saving percentages for the nodes executing the NAS benchmarks over the three power scenarios



(b) The performance degradation percentages for the NAS benchmarks over the three power scenarios



(c) The trade-off distance between the energy reduction and the performance of the NAS benchmarks over the three power scenarios

Figure 10: The experimental results of different static power scenarios

been executed with these two new static power scenarios over the one site one core per node scenario. In these experiments, the class D of the NAS parallel benchmarks were executed over the Nancy site. 16 computing nodes from the three clusters, Graphite, Graphene and Griffon, were used in this experiment.

The energy saving percentages of the NAS benchmarks with the three static power scenarios are presented in Figure 10a. This figure shows that the 10% of static power scenario gives the biggest energy saving percentages in comparison to the 20% and 30% static power scenarios. The small value of the static power consumption makes the proposed scaling algorithm select smaller frequencies for the CPUs. These smaller frequencies reduce the dynamic energy consumption

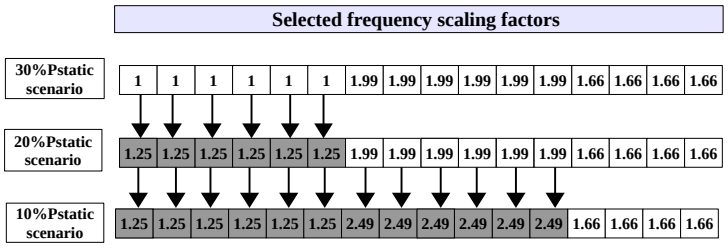


Figure 11: Comparing the selected frequency scaling factors for the MG benchmark over the three static power scenarios

more than increasing the consumed static energy which gives less overall energy consumption. The energy saving percentages of the 30% static power scenario is the smallest between the other scenarios, because the scaling algorithm selects bigger frequencies for the CPUs which increases the energy consumption. Figure 11 demonstrates that the proposed scaling algorithm selects the best frequency scaling factors according to the static power consumption ratio being used.

The performance degradation percentages are presented in Figure 10b. The 30% static power scenario had less performance degradation percentage because the scaling algorithm had selected big frequencies for the CPUs. While, the inverse happens in the 10% and 20% scenarios because the scaling algorithm had selected CPUs' frequencies smaller than those of the 30% scenario. The trade-off distance percentage for the NAS benchmarks with these three static power scenarios are presented in Figure 10c. It shows that the best trade-off distance percentage is obtained with the 10% static power scenario and this percentage is decreased for the other two scenarios because the scaling algorithm had selected different frequencies according to the static power values.

In the EP benchmark, the energy saving, performance degradation and trade-off distance percentages for these static power scenarios are not significantly different because there is no communication in this benchmark. Therefore, the static power is only consumed during computation and the proposed scaling algorithm selects similar frequencies for the three scenarios. On the other hand, for the rest of the benchmarks, the scaling algorithm selects the

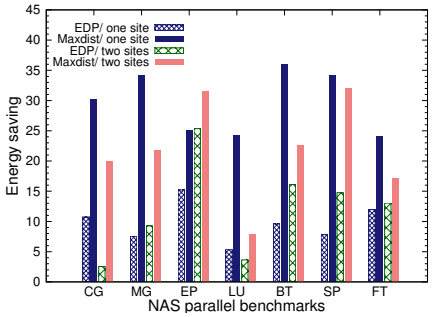
values of the frequencies according to the communication times of each benchmark because the static energy consumption increases proportionally to the communication times.

6.5. Comparison of the proposed frequencies selecting algorithm

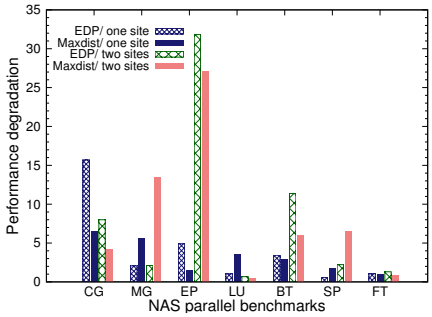
Finding the frequencies that give the best trade-off between the energy consumption and the performance for a parallel application is not a trivial task. Many algorithms have been proposed to tackle this problem. In this section, the proposed frequencies selecting algorithm is compared to a method that uses the well known energy and delay product objective function, $EDP = energy \times delay$, that has been used by many researchers [30, 31, 32]. This objective function was also used by Spiliopoulos et al. algorithm [10] where they select the frequencies that minimize the EDP product and apply them with DVFS operations to the multi-core architecture. Their online algorithm predicts the energy consumption and execution time of a processor before using the EDP method. To fairly compare the proposed frequencies scaling algorithm to Spiliopoulos et al. algorithm, called Maxdist and EDP respectively, both algorithms use the same energy model, Equation 10 and execution time model, Equation 2, to predict the energy consumption and the execution time for each computing node. Moreover, both algorithms start the search space from the upper bound computed as in Equation 18. Finally, the resulting EDP algorithm is an exhaustive search algorithm that tests all the possible frequencies, starting from the initial frequencies (upper bound), and selects the vector of frequencies that minimize the EDP product.

Both algorithms were applied to the class D of the NAS benchmarks running over 16 nodes. The participating computing nodes are distributed according to the two scenarios described in Section 6.2. The experimental results, the energy saving, performance degradation and trade-off distance percentages, are presented in Figures 12a, 12b and 12c respectively.

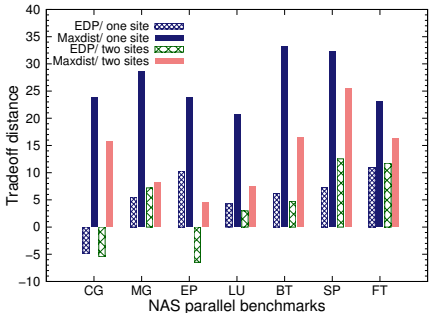
As shown in these figures, the proposed frequencies selection algorithm, Maxdist, outperforms the EDP algorithm in terms of energy consumption reduc-



(a) The energy reduction induced by the Maxdist method and the EDP method



(b) The performance degradation induced by the Maxdist method and the EDP method



(c) The trade-off distance between the energy consumption reduction and the performance for the Maxdist method and the EDP method

Figure 12: The comparison results

tion and performance for all of the benchmarks executed over the two scenarios. The proposed algorithm gives better results than the EDP method because the former selects the set of frequencies that gives the best tradeoff between energy saving and performance. Moreover, the proposed scaling algorithm gives the same weight for these two metrics. Whereas, the EDP algorithm gives sometimes negative trade-off values for some benchmarks in the two sites scenarios. These negative trade-off values mean that the performance degradation percentage is higher than the energy saving percentage. The high positive values of the trade-off distance percentage mean that the energy saving percentage is much higher than the performance degradation percentage. The complexity of both algorithms, Maxdist and EDP, are of order $O(N \cdot M_i \cdot F_j)$ and $O(N \cdot M_i \cdot F_j^2)$

respectively, where N is the number of the clusters, M_i is the number of nodes and F_j is the maximum number of available frequencies. When Maxdist is applied to a benchmark that is being executed over 32 nodes distributed between Nancy and Lyon sites, it takes on average 0.01 ms to compute the best frequencies while the EDP method is on average ten times slower over the same architecture.

7. Conclusion

This paper presents a new online frequencies selection algorithm. The algorithm selects the best vector of frequencies that maximizes the trade-off distance between the predicted energy consumption and the predicted execution time of the distributed applications with iterations running over a heterogeneous grid. A new energy model is used by the proposed algorithm to predict the energy consumption of the application. To evaluate the proposed method on a real heterogeneous grid platform, it was applied on the NAS parallel benchmarks and the class D instance was executed over the Grid'5000 testbed platform. The experiments executed on 16 nodes, distributed over three clusters, showed that the algorithm on average reduces by 30% the energy consumption for all the NAS benchmarks while on average only degrading by 3.2% the performance. The Maxdist algorithm was also evaluated in different scenarios that vary in the distribution of the computing nodes between different clusters' sites or use multi-core per node architecture or consume different static power values. The algorithm selects different vectors of frequencies according to the computations and communication times ratios, and the values of the static and measured dynamic powers of the CPUs. Finally, the proposed algorithm was compared to another method that uses the well known energy and delay product as an objective function. The comparison results showed that the proposed algorithm outperforms the latter by selecting a vector of frequencies that gives a better trade-off between energy consumption reduction and performance.

In the near future, we will adapt the proposed algorithm to take into con-

sideration the variability between some iterations. For example, the proposed algorithm can be executed twice: after the first iteration the frequencies are scaled down according to the execution times measured in the first iteration, then after a fixed number of iterations, the frequencies are adjusted according to the execution times measured during the fixed number of iterations. If the computing power of the system is constantly changing, it would be interesting to implement a mechanism that detects this change and adjusts the frequencies according to the variability of the system. We would like also to develop a similar method that is adapted to asynchronous applications with iterations where iterations are not synchronized and communications are overlapped with computations. 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. Finally, it would be interesting to evaluate the scalability of the proposed algorithm by running it on large platforms composed of many thousands of cores. The scalability of the algorithm can be improved by distributing it in a hierarchical manner where a leader is chosen for each cluster or a group of nodes to compute their scaled frequencies and by using asynchronous messages to exchange the the data measured at the first iteration.

Acknowledgment

This work has been partially supported by the Labex ACTION project (contract "ANR-11-LABX-01-01"). Computations have been performed on the Grid'5000 platform and on the mésocentre of 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.
URL `http://www.top500.org`

[2] U.S. Energy Information Administration, Annual Energy Outlook 2015.
URL http://www.eia.gov/

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

[4] N. B. Rizvandi, J. Taheri, A. Y. Zomaya, Some observations on optimal frequency selection in DVFS-based energy consumption minimization, J. Parallel Distrib. Comput. 71 (8) (2011) 1154–1164. doi:10.1016/j.jpdc.2011.01.004.

[5] J.-C. Charr, R. Couturier, A. Fanfakh, A. Giersch, Dynamic frequency scaling for energy consumption reduction in synchronous distributed applications, in: ISPA 2014, 12th IEEE Int. Symposium on Parallel and Distributed Processing with Applications, IEEE, Milan, Italy, 2014, pp. 225–230. doi:10.1109/ISPA.2014.38.

[6] J.-C. Charr, R. Couturier, A. Fanfakh, A. Giersch, Energy consumption reduction with dvfs for message passing iterative applications on heterogeneous architectures, in: Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International, 2015, pp. 922–931. doi:10.1109/IPDPSW.2015.44.

[7] SimGrid: Versatile Simulation of Distributed Systems.
URL http://simgrid.org

[8] Grid5000.
URL http://www.grid5000.fr/

[9] H. Shen, J. Lu, Q. Qiu, Learning based DVFS for simultaneous temperature, performance and energy management, in: ISQED, 2012, pp. 747–754. doi:10.1109/ISQED.2012.6187575.

[10] V. Spiliopoulos, S. Kaxiras, G. Keramidas, Green governors: A framework for continuously adaptive dvfs, in: International Green Computing Con-

ference and Workshops (IGCC), 2011, pp. 1–8. doi:10.1109/IGCC.2011.6008552.

[11] B. Rountree, D. Lowenthal, S. Funk, V. W. Freeh, B. De Supinski, M. Schulz, Bounding energy consumption in large-scale MPI programs, in: Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on, 2007, pp. 1–9. doi:10.1145/1362622.1362688.

[12] R. Cochran, C. Hankendi, A. K. Coskun, S. Reda, Pack & cap: Adaptive DVFS and thread packing under power caps, in: Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-44, ACM, NY, USA, 2011, pp. 175–185. doi:10.1145/2155620.2155641.

[13] R. Luley, C. Usmail, M. Barnell, Energy efficiency evaluation and benchmarking of AFRL's condor high performance computer, Tech. rep., DTIC Document (2011).
URL http://www.dtic.mil/get-tr-doc/pdf?AD=ADA548738

[14] K. Ma, X. Li, W. Chen, C. Zhang, X. Wang, Greengpu: A holistic approach to energy efficiency in gpu-cpu heterogeneous architectures, in: Parallel Processing (ICPP), 2012 41st International Conference on, 2012, pp. 48–57. doi:10.1109/ICPP.2012.31.

[15] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, Z. Zong, Effects of dynamic voltage and frequency scaling on a k20 gpu, in: Parallel Processing (ICPP), 2013 42nd International Conference on, 2013, pp. 826–833. doi:10.1109/ICPP.2013.98.

[16] N. Muralimanohar, K. Ramani, R. Balasubramonian, Power efficient resource scaling in partitioned architectures through dynamic heterogeneity, in: In Proceedings of ISPASS, 2006. doi:10.1109/ISPASS.2006.1620794.

[17] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C. zhong Xu, A. Zomaya, Energy-aware parallel task scheduling in a cluster, Fu-

ture Generation Computer Systems 29 (7) (2013) 1661 – 1670. doi:10.1016/j.future.2013.02.010.

[18] K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, W. H. Sanders, Blackbox prediction of the impact of DVFS on end-to-end performance of multi-tier systems, ACM SIGMETRICS Performance Evaluation Review 37 (4) (2010) 59–63. doi:10.1145/1773394.1773404.

[19] D. Shelepov, 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. doi:10.1145/1531793.1531804.

[20] D. Li, J. Wu, Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms, Parallel and Distributed Systems, IEEE Transactions on PP (99) (2014) 1–1. doi:10.1109/TPDS.2014.2313338.

[21] J.-J. Chen, K. Huang, L. Thiele, Dynamic frequency scaling schemes for heterogeneous clusters under quality of service requirements, Journal of Information Science and Engineering 28 (6) (2012) 1073–1090.
URL http://www6.in.tum.de/Main/Publications/KHuang2012a.pdf

[22] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, 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, IPDPS '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 4a–4a. doi:10.1109/IPDPS.2005.214.

[23] K. Malkowski, Co-adapting scientific applications and architectures toward energy-efficient high performance computing, Ph.D. thesis, The Pennsylvania State University, USA (2009).

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

[25] J. Zhuo, C. Chakrabarti, Energy-efficient dynamic task scheduling algorithms for dvs systems, ACM Trans. Embed. Comput. Syst. 7 (2) (2008) 17:1–17:25. doi:10.1145/1331331.1331341.

[26] E. Le Sueur, G. Heiser, Dynamic voltage and frequency scaling: The laws of diminishing returns, in: Proceedings of the 2010 Workshop on Power Aware Computing and Systems (HotPower'10), 2010.

[27] N. S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, V. Narayanan, Leakage current: Moore's law meets static power, Computer 36 (12) (2003) 68–75. doi:10.1109/MC.2003.1250885.

[28] T. Rauber, G. Rünger, M. Schwind, H. Xu, S. Melzner, Energy measurement, modeling, and prediction for processors with frequency scaling, The Journal of Supercomputing 70 (3) (2014) 1451–1476. doi:10.1007/s11227-014-1236-4.

[29] NASA Advanced Supercomputing Division, NAS parallel benchmarks (Mar. 2012).
URL http://www.nas.nasa.gov/publications/npb.html

[30] V. Saravanan, A. Anpalagan, I. Woungang, An energy-delay product study on chip multi-processors for variable stage pipelining, Human-centric Computing and Information Sciences 5 (1). doi:10.1186/s13673-015-0046-x.

[31] J. Chen, L. John, Energy-aware application scheduling on a heterogeneous multi-core system, in: Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on, 2008, pp. 5–13. doi:10.1109/IISWC.2008.4636086.

[32] R. Nagpal, Y. Srikant, Exploring energy-performance trade-offs for heterogeneous interconnect clustered vliw processors, in: Y. Robert, M. Parashar, R. Badrinath, V. Prasanna (Eds.), High Performance Computing - HiPC 2006, Vol. 4297, Springer Berlin Heidelberg, 2006, pp. 497–508. doi:10.1007/11945918_48.