Answers to the questions of the reviewers

1 Questions and remarks of the first reviewer

1. In Sec. 5, I found intriguing the fact of executing Algorithm 1 only after the first iteration. I agree with you that your model finds the best tradeoff given some data, but what about the variability? We know computer systems today always show some variability. You are measuring the computation time and energy consumption for one iteration only. Let's suppose something went bad in this first iteration. The scaling factors will not be the best tradeoff because variability has been ignored. What would be the solution for that? Consider variability in the model.

   Answer: In this paper we have considered that the application executes regular iterations over stable computers computing only this application. Therefore, we have assumed that the execution times of all the iterations of the application executed on the same computing node should be almost the same. For this reason we did not take into consideration the variability of the computer system. Moreover, applying the frequency scaling algorithm after many iterations would reduce its impact on the energy consumption especially for applications executing a relatively low number of iterations.

   However, the variability of the computing system can be taken into consideration in a future work. 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.

Taking account of the variability of the system has been added as a perspective at the end of the paper.

2. Another point is that you mention in the abstract and introduction that your solution has low overhead, but it is a centralized solution. Probably it won't scale when we reach hundreds or thousands of computer nodes: take one of that large machines for example. In this paper experiments, only 16 and 32 nodes where considered.

   Answer: We agree with the reviewer that the algorithm is centralized and might be a bottleneck if it was applied to an application running on many thousands of nodes. However, up to 144 nodes in a heterogeneous cluster, the overhead of the algorithm was very small, 0.15 ms, as presented in the simulation results of [6]. We did not execute experiments with more than 32 nodes on Grid'5000 because it does not have many nodes that allow DVFS operations and have energy measurement tools.

   On the other hand, the scalability of the proposed algorithm can be improved if we use asynchronous computations or if the algorithm was distributed in a hierarchical manner where a leader is chosen for each cluster or a group of nodes to compute their scaled frequencies. Improving the scalability of the algorithm is beyond the scope of this paper.

3. In Fig 6, you draw lines between the points. Lines here mean nothing since you are changing the benchmark. I would replot using for instance a non-stacked bar plot with four colors (one site/16, one site/32, two sites/16, two sites/32). I believe it would be much easier to compare and avoid the problem of lines.

   Answer: We agree with the reviewer. The curves in Figures 6 and 8 in the paper were replaced by histograms.

4. About the discussion of results shown in Fig 7, one consideration draws my attention: "(...) 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.". I agree with you that for very large applications, synchronous collective operations are very costly (take a very simple MPI-Allgather for instance). You say that on scale this would produce less energy savings, but your arguments for providing a solution for this was based that today's supercomputers are achieving massive scale.

Answer: In Figure 7, the energy consumption of the benchmarks solving the class D and running on many scenarios are presented. The number of used nodes varies between 16 and 32 in the scenarios while the size of the problem is not modified. Therefore, the computations to communications times ratio is lower when 32 nodes are used instead of 16. When this ratio is small, it means there are not enough computations when compared to the communications times and the impact of scaling down the frequency of the CPU on its energy consumption is reduced. To solve this problem, the problem should be solved on a number of nodes adequate to its size. For example, for the NAS benchmarks, the class E should have been solved on 32 nodes to have a good computations to communications times ratio.

5. In Sec 6.3, why did you choose to keep 32 processes for the evaluation with multi-core clusters? How did you configure MPI for the results

   Answer: In section 6.3, we wanted to evaluate how much energy can be saved when applying the proposed algorithm to message passing applications with iterations running over a grid composed of multi-core nodes. Therefore, the same experiments as in section 6.2 were conducted on the new multi-core platform. Instead of running one process per node as in the previous section, 3 or 4 processes were executed on each multi-core node. The total number of processes, 32 processes, was not modified in order to fairly compare the single core and the multi-core versions.

   Only the architecture file was modified between the single and the multi-core architectures. For the single core architecture, the architecture file contains the name of 32 different nodes. For the multi-core architecture, the architecture file contains less nodes and for every node 3 or 4 slots (cores) are used. The total number of slots is equal to 32.

6. shown in Fig 8a? Some MPI implementations have an option to use shared memory when processes share the same processor. I agree with you in the explanation of the network card utilization, but this shared-memory optimization is possible (sometimes automatically detected by MPI if you pin processes to cores).

   Answer: We did not manually pin processes to cores. Since the communication times increased, we think that the shared memory was not used when two processes, running on the same node, exchange data.

7. In P33, Sec 6.5, you mention that the proposed algorithm outperforms EDP because the former considers both metrics (time, energy) and

the same time. EDP does also, but using a single metric which you have defined: energy x execution time. I think this is only a matter of phrasing.

Answer: We agree with the reviewer, EDP also uses two metrics in the objective function: energy and delay. The sentence in the paper was modified to clarify this misunderstanding. The main difference between our algorithm and the EDP method is the used objective function. For EDP, the product of energy and delay must be minimized, while for our algorithm, the difference between the normalized performance and the normalized energy should be maximized. This new formulation of the objective function allows our algorithm to select the set of frequencies that gives the best tradeoff between the energy consumption and the performance. The objective function of EDP does not give the same frequencies as our algorithm and thus it is outperformed by our method. The results of the experiments confirm that the objective function used by our algorithm is more efficient than the one used by EDP.

8. Other complementary points to consider:

   + P2, L51: there are three dots that looks like an error.

   + P4, L36: also unusual three dots at the end of paragraph.

   + P14 also has three dots in phrase endings. I consider this bad writing style.

   + Fig 2b is missing the X scale ticks. You could show some examples of vectors.

   + P23: "static power is assumed to be equal to 20% of dynamic". Provide citation.

   + Fig 6 is referenced in P23, but appears only in P25. Hard to read.

   + Same for Fig 7.

   Answer: Answer: We have taken in consideration all these remarks and the paper was modified accordingly.

9. From the design of experiments, did you consider using replications? There is no variability metric in your results. Have you run multiple times and got the average (execution time and energy consumption)? I feel that such variability needs to be accounted for, otherwise it is very hard to affirm anything about measurements.

   Answer: Each experiment has been executed many times and the results presented in the figures are the average values of many executions.

Since we have deployed the same operating system on the booked machines and we were the only users executing processes on them during the experiments, no significant variability in the execution time of the applications was noticed.

10. In summary, I think this is a very interesting work but the experimental evaluation lacks variability measurements, consider larger experiments (1K nodes for instance) to see how everything scales, and there is no overhead measurements although authors stress that in abstract/introduction.

    Answer: For the time being, we do not have the resources nor the time to evaluate the proposed algorithm over large platforms composed of more than 1K nodes. However, as said in the perspectives of the paper, the evaluation of the scalability of the algorithm will be in a conducted in a future work as soon as we have access to larger resources. We have discussed the overhead of the algorithm and its complexity in section 6.5 and given in the answer to question 2 some solutions to improve its scalability and reduce its overhead.

    For the variability issue, please refer to the answer to question 1.

2 Questions and remarks of the second reviewer

1. Move the contributions from related work to introduction

2. why emphasize it is a grid platform? the presentation of related work follows the logic of heterogeneous CPUs. Grid is only a type of platform with heterogeneous CPUs

   Answer: The proposed algorithm was adapted to the grid platform which is composed of homogeneous clusters and interconnected by a wide area network which is slower than the local network in each cluster. The algorithm can also work on other heterogeneous platforms.

3. Define what iterative message passing applications are and give exemplar applications of them targeted by this method.

   Answer: The message passing applications with iterations compute the same block of operations several times, starting from the initial solution until reaching the acceptable approximation of the exact solution (we have added this sentence to the paper page 21). There are many example for these applications such as JACOBI, GAUSS-SEIDE, Successive

over-relaxation and conjugate gradient (CG) method and etc. Similarly, an offline method uses DVFS and applied to these applications is in [2]

4. Figure 1 is not clearly explained. Where is the slack time in figure 1 and why slack time =0 for task 1?

   Answer: We agree with the reviewer, this figure was re-plotted to show the slack time. In the figure, we assumed that task 1 is the slower task. So, there are no slack time (waiting time) in the slower task because it is not wait for the others while other tasks wait for it.

5. define the parameters in eq. 1.

   Answer: We have defined Fmax and Fnew in the text.

6. eq. 2: are you assuming each cluster has the same number of nodes?

   Answer: No, we assume each cluster has different number of node, so in the equation we have replaced M_i instead of M and the same for F replaced with F_j in the all equations of the paper.

7. Eq.2 implicitly assumes that there is no overlapping between computation and communication. Is it reasonable?

   Answer: The communications between the computing nodes are synchronized, where each node need to wait for the others to finished their jobs before computing the next iteration. So, there is no overlapping between computations and communications for a node. The overlapping happens when the communications are asynchronous or the computations are not depend on the data sent by the neighbouring nodes.

8. eq. 2 is not clear:

   -how to define and determine the slowest cluster h? the one before scaling or after scaling?

   Answer: The slower task is the task which gives maximum execution time before scaling the frequency of the node. We have added this sentence to the paper (page 8).

   - what is the communication time without slack time

   Answer: There is no synchronous communications with zero slack times, but if a node send a message to another node which is already waiting for that message. The latter will acknowledge the reception of the message from the sender without any delay. On the other hand, if the

receiving node is still computing the sender has to wait for it to finish its computation to acknowledge the reception of the message. This time is called the slack time.

- in equation, min operation is used to get the communication time, but in text, it says to use the slowest communication time, which should use the max operation then.

Answer: We agree with the reviewer and the sentence "slower communication time" changed to "communication time of the slower node" in the paper.

9. discuss the difference between eq. 2 and the prediction model in references [5] and [6]

   Answer: The prediction models in [5] and [6] are for homogeneous and heterogeneous clusters respectively, while eq. 2 is for a grid. where the homogeneous cluster predication model was used one scaling factor denoted as S, because all the nodes in the cluster have the same computing powers. Whereas, in heterogeneous cluster prediction model all the nodes have different scales and the scaling factors have denoted as one dimensional vector $(S_1, S_2, \ldots, S_N)$. The execution time prediction model for a grid Equation (2) defines a two dimensional array of scales $(S_{11}, S_{12}, \ldots, S_{NM_i})$. We have added this to the paper (page 8).

10. Eq. 10: Can the authors comment on the energy consumed by communications?

    Answer: The CPU during communications consumed only the static power power. While in computations the CPU consumes both the dynamic and static communication, refer to [1]. We have added this sentience to the paper, page 11.

11. This work assume homogeneous cpu in one cluster. Line 55 says: even if the distributed message passing iterative application is load balanced, the computation time of each cpu j in cluster i may be different Why?

    Answer: The computation times may be slightly different due to the delay caused by the scheduler of the operating system. We have added this in the paper.

12. Comment why the applications in NAS parallel benchmark are iterative application? These applications are normally run in one cluster. Describe in more detail how they are run across multiple clusters.

Answer: The applications in NAS parallel benchmark are application with iterations because they iterate the same block of instructions (communications and computations) many times. All the benchmarks are MPI programs that allowed to be executed on any distributed memory platform such as clusters and grids with no required modifications. Since, we have deployed the same operating system on all the nodes, we just compile the source on one cluster and then copied the executable program on all the clusters.

13. broken sentence in line 28 on page 12

    Answer: The word "were" replaced with "where".

14. Why T_{old} is computed using eq. 12, which applies MAX over computation time and communication time, while in T_{new}, max and min operations are applied over computation and communication separately?

    Answer: We agree with the reviewer, T_{old} is the maximum execution time of the application before scaling the frequency and it is computed as in T_{new} equation without scaling factors. So, we have changed the T_{old} in the paper as as follows:

$$T_{old} = \max_{\substack{i=1,2,...,N \\ j=1,2,...,M_i}} (T_{cp_{ij}}) + \min_{i=1,2,...,N} (T_{cm_{hj}}) \quad (1)$$

15. Line 55 on page 16 is to define the slack time, which should be introduced at the beginning of the paper, such as in figure 1.

    Answer: We have changed it in the paper and added to page 6.

16. Authors comment whether (and how) the proposed methods can be applied/extended to other programming models and/or platform, such as mapreduce, heterogeneous cluster with CPU+GPU. Revision

    Answer: The proposed method can only be applied to parallel programming with iteration and with or without message passing. Indeed, the proposed method can be applied to the parallel application with mapreduce if it is a regular application with iterations. Therefore, the time of each map and reduce operations (communications) and the computation times in the program must be computed at the first iterations to predict the energy consumption and the execution time. After, the proposed algorithm can be used as it to select the best frequencies. The proposed method can be applied to a heterogeneous platform composed from GPUs and CPUs, since modern GPUs like CPUs allow the use of DVFS operation.

3 Questions and remarks of the third reviewer

1. suggest the authors to use much larger size of nodes, instead of on 16 nodes, distributed on three clusters, to see the scalability of the energy saving

   Answer: We have made the experiments not only on 16 nodes, but we have also made them over 32 nodes distributed over three clusters and in the near future we will apply the proposed method over a larger number of nodes.

2. the energy saving is actually calculated by the quantitative formula instead of the real measurements. Can you have any discussions on the real measurements?

   Answer: The scope of this paper is not mainly focuses on the energy measurements, but it focuses on modelling and optimizing the energy and performance of grid systems. The proposed energy model depends on the dynamic and static power values for each CPU. We have used a real power measurement tools allowed in Grid'5000 sites to measure the dynamic power consumption. Moreover, the real measurements are difficult for a grid platform when the nodes are geographically distributed. As a future work, it is interesting to compare the accuracy of the proposed energy model with a real instruments to measure the energy consumption for local clusters such as the measurement tools presented in [3].

3. the overhead is not measured, can you present something on this as well to demonstrate what the authors claimed "has a small overhead and works without training or profiling"?

   Answer: In the comparison section 6.5, we have presented the execution time of the algorithm when it is executed over 32 nodes distributed over three sites located at two different sites, it takes on average 0.01 ms. The algorithm works online without training which means it only uses the measured communication and computation times during the runtime and do not require any profiling or training executed before runtime.

References

[1] Vincent W. Freeh, Feng Pan, Nandini Kappiah, David K. Lowenthal, and Rob 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, pages 4a–4a, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Amina Guermouche, Nicolas Triquenaux, Benoît Pradelle, and William Jalby. Minimizing energy consumption of MPI programs in realistic environment. *CoRR*, abs/1502.06733, 2015.

[3] Gustavo Rostirolla, Rodrigo Da Rosa Righi, Vinicius Facco Rodrigues, Pedro Velho, and Edson Luiz Padoin. Greenhpc: a novel framework to measure energy consumption on hpc applications. In *2015 Sustainable Internet and ICT for Sustainability, SustainIT 2015, Madrid, Spain, April 14-15, 2015*, pages 1–8. IEEE, 2015.