An Energy-aware Multi-start Local Search Heuristic for Scheduling VMs on the OpenNebula Cloud Distribution

Yacine Kessaci, Melab Nouredine, El-Ghazali Talbi

An Energy-aware Multi-start Local Search Heuristic for Scheduling VMs on the OpenNebula Cloud Distribution

Yacine Kessaci, Nouredine Melab
and El-Ghazali Talbi
INRIA Lille, CNRS/LIFL, Université Lille 1.
Parc Scientifique de la Haute Borne,
40 avenue Halley Bât.A, Park Plaza,
59650 Villeneuve d'Ascq, France.
Email: {yacine.kessaci, nouredine.melab, talbi}@lifl.fr

***Abstract*—Reducing energy consumption is an increasingly important issue in cloud computing, more specifically when dealing with a cloud distribution dispatched over a huge number of machines. Minimizing energy consumption can significantly reduce the amount of energy bills, and the greenhouse gas emissions. Therefore, many researches are carried out to develop new methods in order to consume less energy. In this paper, we present an Energy-aware Multi-start Local Search algorithm for an OpenNebula based Cloud (EMLS-ONC) that optimizes the energy consumption of an OpenNebula managed geographically distributed cloud computing infrastructure. The results of our EMLS-ONC scheduler are compared to the results obtained by the default scheduler of OpenNebula. The two approaches have been experimented using different (VMs) arrival scenarios and different hardware infrastructures. The results show that EMLS-ONC outperforms the previous OpenNebula's scheduler by a significant margin in terms of energy consumption. In addition, EMLS-ONC is also proved to schedule more applications.**

***Index Terms*—energy-aware scheduling, cloud distribution, cloud computing, resource allocation, OpenNebula, multi-start, local search.**

I. Introduction

The field of cloud computing uses different management techniques for data center virtualization such as OpenNebula [1]. However, computers use a significant and growing portion of energy in the world specifically when dealing with distributed cloud computing infrastructure. Therefore, energy-aware computing is crucial for large-scale systems that consume considerable amount of energy.

A recent study [2] shows that, the power used by servers represents about 0.6% of total U.S. electricity consumption. That proportion grows to 1.2% when cooling and auxiliary infrastructures are included. In the same year, the aggregate electricity bill for operating those servers and associated infrastructure was about $2.7 billions and $7.2 billions for the U.S. and the world, respectively. The total electricity consumed by servers doubled over the period 2000 to 2005 in worldwide and this increase was further confirmed in the last 5 years (2005-2010) [3].

In addition, according to an Amazone's estimate [4], the energy-related costs amount represents 42% of the total data center budget, and includes both direct power consumption 19% and cooling infrastructure 23%, these values are normalized with a 15 years amortization. It clearly appears that all the issues cited before are important to tackle and represent a huge challenge.

In this paper, we present a new work that aims to deal with the energy consumption within a realistic cloud infrastructure using OpenNebula as a software management solution. Indeed, we propose a scheduler instead of the one that is currently used by OpenNebula. Our scheduler is based on a multi-start local search heuristic that helps to find the best scheduling by dispatching the arriving of virtual machines (VM) according to the minimum energy consumption. A virtual machine is a software based machine emulation technique to provide a desirable, on demand computing environments for users. This approach uses information provided by the hypervisor on each host to find the best scheduling since the energy consumption is straightly related to the general host features and to its current hardware usage information. Our approach aims also to give the best Quality of Service (QoS) which consists in satisfying the maximum number of client by meeting their VMs requirements. The multi-start makes it possible by exploring a wide range of potential solutions to the problem.

The remainder of the paper is organized as follows. In Section II we present the related work to our approach. Section III presents the application, system and energy models used in our problem modeling. Our approach is presented in Section IV. The results of our experimental study are discussed in Section V. The conclusion is drawn in Section VI.

II. Related Work

After a race to performance, utility and cloud computing paradigm are facing an energy problem. Hence, several works have been proposed in the field of the energy aware computing. However, most of those approaches tackle this topic by referring and focusing on scheduling dedicated applications. In [5],

[6] for example a hardware technique (DVFS) is proposed. It consists of varying the CPU frequency in order to minimize the energy consumption. The drawback of this type of methods is the assumption that they make about a tight coupling between the tasks and the resources. Another way of reducing cloud computing energy footprints is proposed in [7]. This work uses the potential offered by the virtualization in order to apply a task consolidation through two heuristics in order to maximize the resource utilization. In [8] the author presents a reinforcement learning approach to deal with the optimization of two main aspects, performance and power consumption. All the previous presented works aim to reduce the energy consumption on single data centers or on multiple servers geographically concentrated. Except the work proposed in [9] which deals with energy consumption reduction in large-scale computational grids like Grid5000, by switching off idle nodes in a clever way. Besides, the work presented by *Kessaci et al* in [10] deals with the energy consumption, gas emissions and pricing objectives of a geographically distributed cloud infrastructure using a Pareto multi-objective genetic algorithm. The work in [11] proposes a VM scheduling algorithm based on DVFS technique to reduce the energy consumption of a single OpenNebula virtualized cluster. The idea behind this work is to reduce the clock frequency of the cluster as low as possible to fit exactly the VMs requests.

The major difference of our work with the approach [11] is that our work deals with not only one cluster but a large number of geographically distributed machines. Indeed, our approach tends to provide a scheduler on an HPC virtualized environment.

In addition, all of the last presented approaches tackle the energy minimization issue by experimenting their approaches via simulation. The only work that uses a real infrastructure for validation, is the one proposed in [9] by using Grid5000. The problem with this work is that it has been conduced on a grid and not on a cloud using virtual machines.

To deal with all the misses mentioned before, we propose a scheduler using a multi-start local search to optimize the energy on a distributed cloud infrastructure managed by OpenNebula. Therefore, in our approach we integrated our scheduler instead of the default one of OpenNebula. This was possible, given that the software is open source. Hence, our approach can be deployed at the same time with OpenNebula on a real distributed infrastructure to schedule VMs.

III. Distributed Cloud Scheduling Model

A. System Model

Our model is based on an Infrastructure As A Service (IAAS) cloud model working under an OpenNebula cloud manager. Indeed, we are dealing with a two-tier architecture with in each side respectively a distributed cloud provider and clients. These latter have access to the cloud by requesting resources to the provider. The service proposed by the cloud provider in our approach is offering VMs to the clients in order to compute their HPC applications. The role of this work is to help the provider to optimize two criteria while proposing its service. The model of our cloud is geographically distributed over the world. The originality of this approach is to propose a scheduling algorithm that uses a multi-start local search in order to find the best scheduling to the VMs on the hosts which composes the cloud. The location of the default OpenNebula's scheduler and its replacement by our EMLS-ONC scheduler is shown in Figure 1. The objectives considered in our algorithm are the energy and the number of satisfied clients (QoS). The scheduler algorithm aims to give the best Quality of Service (QoS) by allocating the maximum number of VMs, while helping the provider to minimize his energy consumption. The optimization of the objectives is due to the heterogeneity of the hosts that compose the distributed cloud. The heterogeneity means different cpu, memory and storage capacities. It means also different cpu speeds and different loads on each host. This disparity has the same property as the DVFS technique, since it provides different frequencies without needing well-equipped machines.

B. Energy Model

The energy consumption of a data center results from IT equipments and auxiliary equipments. In our work we consider cooling energy consumption among the auxiliary equipment and computing energy consumption in IT equipments. Indeed, since our approach treats about HPC usages on a virtualized environment, the most of the energy consumption is resulted from the intensive computation.

Our processors energy model is derived from the power consumption model in Complementary Metal-Oxide Semiconductor (CMOS) logic circuits given by $P = \alpha f^3 + \beta$.

In addition, another source of energy consumption needs to be taken into account. In fact, the energy used for cooling the data centers is consequent and has to be integrated in our energy model. Energy dedicated to cooling is tightly related to the load of each host. Indeed, the more a host is loaded the higher its processor frequency is and the more the cooling system is utilized. The scheduler is informed by the hypervisor of each host about the current frequency of its processors.

C. Problem Description

Our problem is composed of two-tier architecture. The first tier is a cloud provider which has N hosts (data centers) geographically distributed. The second tier is clients with J VMs requests for running HPC applications. The problem consists of scheduling J applications on N data centers. We know that the task scheduling problem in general is NP-hard [12]. Therefore our VMs scheduling problem is NP-hard as well. Thus, a metaheuristic algorithm (Local search) appears to be the most appropriate approach to adopt.

In the normal OpenNebula virtual machine features, the client submits virtual machine requests with QoS requirements. Those requirements are the number of cpu and their speeds, memory size, storage capacity, the type of the operating system, etc. In our problem, we added a time requirement in the definition of the VM to know the duration of the request.

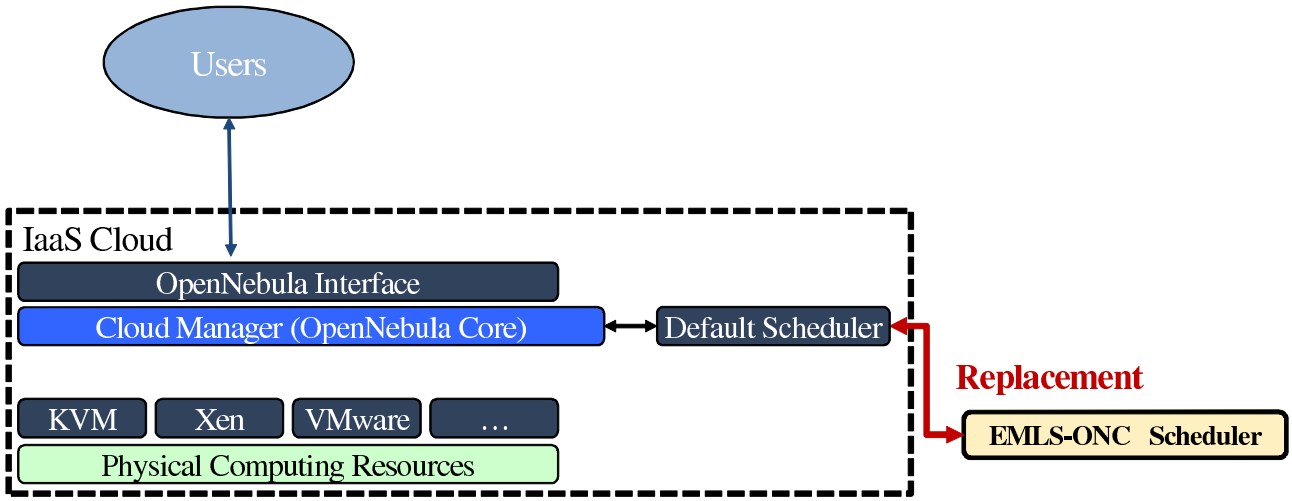

Fig. 1. How to Replace the Scheduler of OpenNebula with the EMLS-ONC Scheduler in a Distributed IAAS Cloud

Therefore, during the scheduling process, the user submits a request for a VM j. A VM in our model is defined by a triplet (e_j, n_j, m_j), all the triplet information are given by the user during the submission, except the starting time of the VM (t_j) which is deduced from the submission time. The elements of the triplet represent the duration runtime of the VM (e_j), the number of processors needed by the user for his VM (n_j) and finally the memory size (m_j). Our triplet is inspired from Amazon EC2 [13] which asks the user to give the duration time of his reservation. Thus, the user has sometimes to pay for a longer VM reservation to ensure the completion of the applications that he/she runs on it even if this latter finishes before the end of the reservation time.

The objective functions of our approach aim to minimize the energy consumption of the entire infrastructure and to provide the best QoS. It is formulated as follows:

$$Minimizing\ the\ energy\ consumption = \sum_{i}^{N}\sum_{j}^{J}(E)_{ij} \quad (1)$$

Where $(E)_{ij}$ is the power consumption of the host i while be used by the VM j. This is done always by respecting the following constraints:

- Each VM j has to find at least one host with the correct requirements to be assigned on it otherwise the VM is rejected.
- Each VM j can be assigned to one and only one host i.

IV. Multi-start Local Search Algorithm for VMs Scheduling

A. Problem Encoding

In order to formulate our problem without overriding the previous constraints (i.e. The VM has to find a host with its requirements and each VM can be scheduled only on one host), we propose an encoding for the EMLS-ONC solutions (see Figure 2).

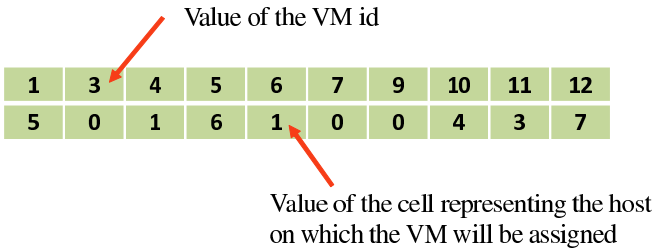

Fig. 2. Problem Encoding

Figure 2 represents one possible scheduling among plenty that proposes the multi-start local search algorithm. In the proposed example we identify three major specifications. The values of the first row of the table (map keys) depict the VMs that are scheduled, the number which is contained by each cell of the second row of the table identifies the host to which the VM is allocated. In other words, if we look at Figure 2, the first column represents the first VM of the pool that is currently treated by the EMLS-ONC, it is identified with the id 1. This VM is allocated to the host 5. The second VM with the id 3 is allocated to the host 0 and so on. This encoding tells us about the number of VMs contained in the pool, our example contains 10 VMs. This encoding helps also us to deal with the characteristics of our problem. Indeed, it allows scheduling all the VMs of the pool, each VM will be assigned to one and only one host (no duplication of map keys). A host can handle more than one VM and not all the hosts are necessarily used in each solution.

B. Population Initialization

The generation of the initial solution in a local search algorithm is an important phase. In fact, this step affects the future results quality. In our approach, since we deal with a multi-start method, each local search execution has

its own initialization and then its proper initial solution. This process follows a random and greedy method. Indeed, after the common phase of the hosts filtering (removing the hosts with not the correct requirements), each VM obtains a set of hosts on which it can be assigned. The first phase of the initialization consists in picking a host from the host set of each VM randomly. For each host selection, a checking mechanism is done to verify the respect of the constrains and the availability of the host in case where previous VMs in the solution already used the resources. If the host is not available any more, the greedy phase gets involved and assigns the VM to the next host in the host set. If no hosts are available for the VM, this latter is removed from the current scheduled pool and will be scheduled during the next VM arrival wave.

C. Scheduling Steps

Before each scheduling, the scheduler waits for a fixed period of time called *scheduling cycle*. This period helps to gather a pool of VMs in order to have a larger choice and thus to optimize the future scheduling. Once this phase done the host pool is filtered out to keep only the hosts with the correct requirements. The multi-start phase launches each local search algorithm separately. The number of launched LS is equal to the minimum value between the number of hosts composing the distributed cloud and 20. The reason of this choice is due to relationship between the complexity of the problem and the number of hosts. Indeed, a small number of hosts makes easier the scheduling since it reduces the opportunities of VMs assignments. Therefore, few local searches are enough to get a good solution. However, the drawback of relating the number of launched local searches to the number of hosts is the time consumption. A tradeoff has been found by bounding their number to 20. After the end of each local search process, all the best solutions of each LS are compared. Only the best solution among all the LSs solutions is kept and chosen to be the scheduling. In the last step, OpenNebula dispatches the VMs according to this solution, it updated the hosts states and a new scheduling cycle is started. All the scheduling steps are drawn in Figure 3.

D. EMLS-ONC Algorithm

The role of the local search algorithm is to make a number of combinations from the initial solution using neighborhood relation in order to find the best scheduling according to the specified objective. Using the multi-start adds diversity, while each local search plays the role of intensification. The local search algorithm starts by generating the initial solution. The initialization process is explained in Section IV-B. This initial solution is used to generate a neighborhood based on two neighborhood operators. Both operators are based on an exchange process. The first operator is dedicated to generate neighborhoods for small cloud configuration (less than 50 hosts), while the second is dedicated to big neighborhoods with huge distributed infrastructure. The first operator switches the value of the host of each VM in the initial solution with each value of the VM's set of hosts exhaustively. In the second operator, the number of hosts is big. Therefore, the algorithm can not afford to enumerate all the hosts in a reasonable time. Thus, it switches the selected host with not all the VM's hosts set but only with a randomly selected range among this set. In other words, one iteration of the LS, to generate one neighborhood, represents the enumeration of all the hosts in the set of each VM, or the enumeration of all the hosts of the selected range case of a second operator usage. Each solution of the generated neighborhood is checked for its feasibility. A fitness value is also assigned to this solution. The best solution of the neighborhood is kept to build another neighborhood during the next iteration using the previous operators. The algorithm stops when the number of iterations reaches the number of VMs. Like the number of local searches in the multi-start, the reason of choosing this value is related to the complexity of the problem. Indeed, the more VMs they are the more iterations are needed to find a good solution.

V. Experiments and Results

This section presents the results obtained from our comparative experimental study. The experiments aim to demonstrate and evaluate the contribution of the multi-start local search approach compared to the default OpenNebula's scheduler.

A. Experimental Settings

The experimental settings concern both sides of our model, client side with its VMs and provider side with the hardware configuration of the cloud.

- **VMs' settings:** We generated VM in XML format to let the OpenNebula parser read their features and be as realistic as possible. The VM features in our experiments vary according to three points as said before in Section III-C with the triplet (e, n, m), in order to fit the algorithm parameters. Therefore, we generated randomly the execution time e from [1,10] hours, the processor requirement n from [0.5,8] and finally the memory needs m from [1,3] GBs.
- **Distributed cloud settings:** As for the VMs the hosts features are provided in XML format to the OpenNebula parser. Hence, we generated different types of hosts by changing each time their features. Each host is specified by its number of cores randomly generated between [1,24] cores, its memory capacity from [2,24] GB, its CPU speed from [1,3] Ghz and finally the number of VMs already running on it [0,10]. In addition, the host features are logical. Indeed, a host has also information about its usage rate, which is randomly generated and never exceed the initial capacity. The free resources are deduced from the initial host capacity and its current usage rate.

B. Algorithm and Experiments Parameters

Our approach deals with VMs for HPC usage. Therefore, we conducted our experiments on only one scheduling cycle. Indeed, this experiment protocol helps to compare the ability of both algorithms EMLS-ONC and the default OpenNebula scheduler to handle a big number of VMs requests at the same

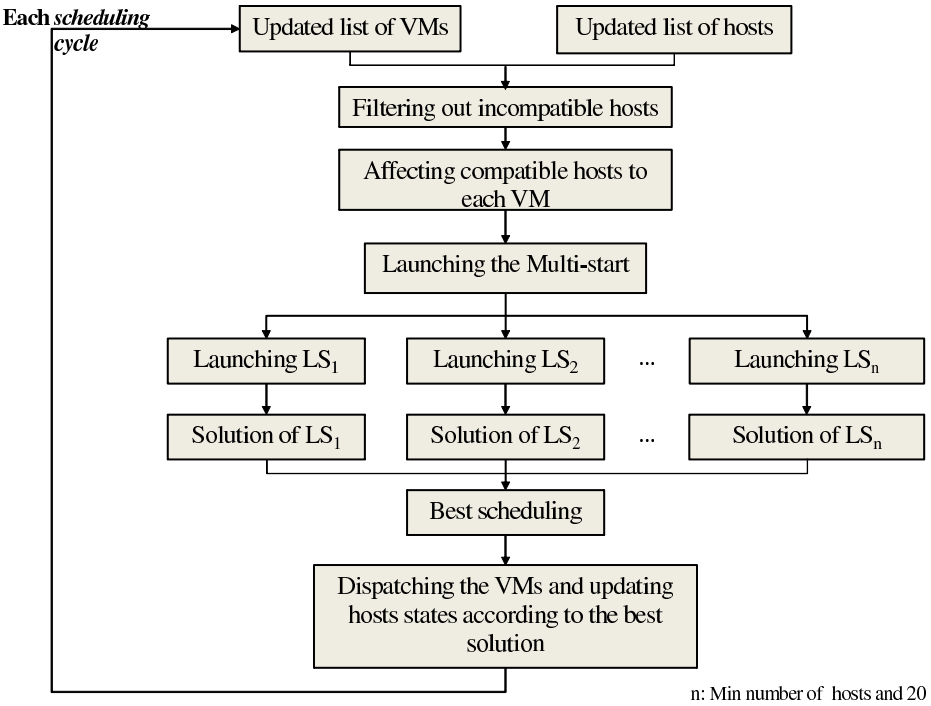

Fig. 3. The Flowchart of the EMLS-ONC Algorithm

time. In addition, comparing them on one *scheduling cycle* helps to get their processing time. In fact, both algorithms have as a constraint to provide their results before the end of the scheduling cycle time, and so the arrival of a new pool of VMs. Moreover, the experiments deal with the scheduling process part of the algorithm not the physical dispatching of the VMs.

In our experiments we used some parameters (Scheduling cycle, number of VM per host at each scheduling cycle, varying arrival rate of the VMs and number of hosts composing the cloud). We performed experiments with 4 different cloud configurations, from a small local cloud with 5 hosts to a massive geographically distributed cloud with 1280 machines. Concerning varying arrival rates we use 4 different VMs loads from a single VM to a massive arrival of 100 VMs on one scheduling cycle. We also limited the number of assigned VM per host for each scheduling cycle to 3 VMs. A full description of the experiments parameters is given in Table I.

TABLE I
EXPERIMENTAL PARAMETERS

Parameter	Value
Max VM per host	3
Scheduling cycle	30s
Number of VM per arrival rate	1, 10, 40, 100
Number of hosts	5, 80, 320, 1280

C. Experimental Results

To the best of our knowledge no previous approach deals with a multi-start local search scheduler for a realistic OpenNebula managed distributed cloud. Thus, we perform a set of experiments with different parameters. In addition of optimizing the energy, the approach has first to satisfy the maximum number of clients (VMs requests). A comparison between our approach and the previous OpenNebula's scheduler appears to us to be the best choice to evaluate our work. Since our method is a metaheuristic, it has a part of stochasticity. Therefore, we run each experiment 20 times. The results are the average value of all the executions. They are presented in Table II. Experiments show that our approach improves the results obtained by the OpenNebula scheduler in all the configuration while scheduling averagely more VMs. There is only two configurations where the results obtained by the previous OpenNebula's scheduler are better than the EMLS-ONC ones. This phenomenon can be explained for the first instance (10 VMs, 80 hosts) by the lack of intensification. Indeed, as said before the number of LS iterations is related to the number of VMs. In the other instance (40 VMs, 5 hosts), the problem is due to the lack of diversification caused by the small number of LS starts. Hence, the solution gets trapped in a local optima despite the intensification.

In addition, EMLS-ONC returns results in a small time duration, largely lower than the threshold represented by the scheduling cycle. It respects therefore, the rapidity required for schedulers.

To summarize, in our approach, the energy consumption is reduced averagely by **15,6%** compared to the OpenNebula default scheduler, while assigning averagely **1,3%** more VMs than this latter.

TABLE II
COMPARISON BETWEEN THE RESULTS OBTAINED BY THE EMLS-ONC ALGORITHM AND THE OPENNEBULA SCHEDULER

Number of VMs	1					
Number of hosts	Number of scheduled VMs		Consumed Energy (Energy Unit)		Time (s)	
	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler
5	**1**	1	**453035**	453035	**10e-04**	10e-06
80	**1**	1	**2505407.9**	3973660	**10-e03**	10e-06
320	**1**	1	**2841140.6**	3973660	**10-e03**	10e-06
1280	**1**	1	**181747.4**	226517	**10-e02**	10e-06

Number of VMs	10					
Number of hosts	Number of scheduled VMs		Consumed Energy (Energy Unit)		Time (s)	
	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler
5	**6**	7	**3328294.6**	3624280	**10e-03**	10e-06
80	**10**	10	**22162200**	19340900	**10e-02**	10e-05
320	**10**	10	**22108773.3**	34809800	**10e-02**	10e-02
1280	**10**	10	**12144173.3**	13509200	**0.2**	10e-04

Number of VMs	40					
Number of hosts	Number of scheduled VMs		Consumed Energy (Energy Unit)		Time (s)	
	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler
5	**11.6**	12	**25837606.7**	18146400	**10e-02**	10e-04
80	**40**	40	**56091493.3**	73667200	**0.1**	10e-04
320	**40**	40	**68386173.3**	84563000	**0.2**	10e-04
1280	**40**	40	**59823060**	65303900	**0.8**	10e-04

Number of VMs	100					
Number of hosts	Number of scheduled VMs		Consumed Energy (Energy Unit)		Time (s)	
	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler	EMLS-ONC	OpenNebula scheduler
5	**13.9**	14	**3100997.3**	41853300	**10e-02**	10e-04
80	**91.7**	84	**175029600**	181780000	**0.7**	10e-03
320	**100**	100	**132239933**	149678000	**1**	10e-03
1280	**100**	100	**135882200**	161002000	**2.4**	10e-03

VI. CONCLUSION

In this paper, we presented a new scheduler for the cloud manager OpenNebula using a multi-start local search algorithm to minimize the energy consumption, while satisfying the clients QoS by assigning the maximum VMs. The energy saving of our approach exploits the disparity and the difference in the features of the hosts that compose the distributed cloud.

Our new approach has been evaluated with XML generated pools of VMs and hosts to fit the OpenNebula specifications since EMLS-ONC is embedded in this latter. Experiments show that our multi-start LS improves on average the results obtained by the OpenNebula's default scheduler by **15,6%**. In addition, our approach schedules on average **1,3%** more VMs than the OpenNebula's previous scheduler.

Besides, the major perspectives of this work is to minimize more the energy consumption by using a better energy model including other energy consumption resources like memory and hard drives. Moreover, the model will be improved by adding other objectives like green house gas emission, profit, etc. In addition, we can imagine a dynamic EMLS-ONC scheduler which will reassign VMs during their running phase on different hosts to optimize more the energy. However, this will depend on the flexibility, the data transfer cost and the CPU time complexity of the VMs since we deal with HPC applications VMs.

Finally, we are planning to deploy our algorithm through the OpenNebula cloud distribution, on top of the geographically distributed cloud offered by the EGI grid infrastructure. Thus, we will give the opportunity to exploit the European geographical dispersion offered by EGI for energetic and / or environmental purposes.

REFERENCES

[1] J. Fontan, T. Vazquez, L. Gonzalez, R. S. Montero, and I. M. Llorente, "Opennebula: The open source virtual machine manager for cluster computing," in *San Francisco, CA, USA*, May 2008.

[2] J. G. Koomey, "Estimating total power consumption by servers in the U.S. and the world."

[3] (2011) Efficape energie. http://www.efficap-energie.com/.

[4] J. Hamilton, "Cooperative expendable micro-slice servers (CEMS): Low cost, low power servers for internet-scale services," in *Proceedings of 4th Biennial Conference on Innovative Date Systems Research (CIDR), Asilomar, California, USA, January*, 2009.

[5] Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in

CCGRID'09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 92–99.

[6] N. B. Rizvandi, J. Taheri, A. Y. Zomaya, and Y. C. Lee, "Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 388–397, 2010.

[7] Y. Lee and A. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, pp. 1–13, 2010, 10.1007/s11227-010-0421-3.

[8] G. Tesauro, R. Das, H. Chan, J. O. Kephart, D. Levine, F. L. R. III, and C. Lefurgy, "Managing power consumption and performance of computing systems using reinforcement learning," in *NIPS*, 2007.

[9] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas, "Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems," in *Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on*, 2008, pp. 171 –178.

[10] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based GA for scheduling HPC applications on distributed cloud infrastructures," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, july 2011, pp. 456 –462.

[11] *Power-Aware Scheduling of Virtual Machines in DVFS-enabled Clusters*. New Orleans, LA: IEEE Computer Society, 08/2009 2009. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/09-greenit-cluster09/vonLaszewski-cluster09.pdf

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[13] Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/fr/ec2/.