Energy Aware Clouds Scheduling Using Anti-load Balancing Algorithm - EACAB

Cheikhou Thiam, Georges Da-Costa and Jean-Marc Pierson
Institut de Recherche en Informatique de Toulouse, Universite de Toulouse 3 Paul Sabatier, 118 Route de Narbonne, Toulouse, France

Keywords: Energy, Virtual Machines, Cloud, Consolidation.

Abstract: Cloud computing is a highly scalable and cost-effective infrastructure for running HPC, enterprise and Web applications. However rapid growth of the demand for computational power by scientific, business and web-applications has led to the creation of large-scale data centers consuming enormous amounts of electrical power. Hence, energy-efficient solutions are required to minimize their energy consumption. The objective of our approach is to reduce data center's total energy consumption by controlling cloud applications' overall resource usage while guarantying service level agreement. This article presents Energy aware clouds scheduling using anti-load balancing algorithm (EACAB). The proposed algorithm works by associating a credit value with each node. The credit of a node depends on its affinity to its jobs, its current workload and its communication behavior. Energy savings are achieved by continuous consolidation of VMs according to current utilization of resources, virtual network topologies established between VMs and thermal state of computing nodes. The experiment results show that the cloud application energy consumption and energy efficiency is being improved effectively.

1 INTRODUCTION

Up to now, the problem of efficiently allocating tasks in clusters has received considerable attention. Task scheduling algorithms have been proposed to optimize the placement of tasks with respect to performance-related criteria. Usually, those researches do not use migration to consolidate workload.

In recent years, many authors have studied the problem of power aware placement, finding theoretical solutions as well as practical ones(Lawson and Smirni, 2005)

In a similar field of research, performance of distributed systems, load-balancing techniques are often used in order to guaranty good performance. These techniques work by spreading load on all available servers, which is efficient from the performance point of view, but not from the energy point of view.

Typical performance measures include task response time, throughput and processor utilization. But when the goal is to reduce energy consumption, this type of algorithms can lead to have hosts largely under-loaded and therefore consuming energy unnecessarily.

In this context, this article proposes an energy-aware anti-load balancing algorithm. This run-time algorithm will migrate tasks while they are running, using on-the-fly migration technology.

Classical *anti-load balancing* algorithms based on migration techniques(Thiam and Da Costa, 2011) use mainly one single parameter: Threshold. Depending on a server load, if over a threshold, load is migrated on other servers or a new server is switched-on in order to keep a good quality of service. If under a threshold, load is migrated on other servers and the host is switched off.

Most researches on energy efficiency try to reduce energy consumption of servers, but they usually do not take into account the cost of cooling systems and related infrastructure. Thus it is important to set a minimum threshold to consolidate tasks, but also to avoid to load hosts heavily by concentrating too many tasks on the same host.

The main contribution of this work is to propose an efficient algorithm that migrate tasks to reduce energy-consumption while preserving performance and preventing hot spots. The algorithm, called EACAB, is based on results from the field of load balancing, especially on ALB (Thiam and Da Costa, 2011)

Thiam C., Da-Costa G. and Pierson J..
Energy Aware Clouds Scheduling Using Anti-load Balancing Algorithm - EACAB.
DOI: 10.5220/0004856600820089
In *Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems* (SMARTGREENS-2014), pages 82-89
ISBN: 978-989-758-025-3

algorithm. We evaluate EACAB by simulation using Enersim.

The rest of the paper is organized as follows. Section 2 discusses related work, followed by the model in Section 3. The proposed migration algorithms are discussed in Section 4. An analysis of simulation results of the proposed algorithms is presented in Section 5.

2 RELATED WORK

In this section, we present energy consumption models and migration techniques.

2.1 About Energy Consumption

In the past few years, people start to realize that the energy consumption is a critical issue since energy demand of clusters have been steadily growing following the increasing number of data centers. Several strategies for energy saving in heterogeneous clusters have been proposed and studied. Recently, many energy-aware scheduling algorithms have been developed primarily using the dynamic voltage-frequency scaling (DVFS) capability which has been incorporated into recent commodity processors. However, these techniques are rarely compatible with optimizing both quality of service for tasks and energy consumption.

Chase et al. (Chase et al., 2001) illustrated a method of determining the aggregate system load and the minimal set of servers that can process a load. A similar idea leverages work in cluster load balancing to determine when to turn machines on or off to handle a given load (Pinheiro et al., 2001). A critical problem for these ideas is that in order to turn lightly loaded machines off or to assign workload to newly turned-on machines, the task need to be transferred from one machine to another. But almost all the operating systems used in the real clusters, e.g. Windows, Unix and Linux, cannot support such kind of operations. So in their research specific OS features have to be developed and applied, which in turn limits the practicability of their approaches. Research efforts are in great need to architect green data centers with better energy efficiency. The most prominent approach is the consolidation enabled by virtualization. Server virtualization technology has become known to improve utilization while reducing various aspects of consumption and offering the ability to ride the *green* trend while helping businesses save money. There are indeed a few high-performance computers designed with energy-saving in mind, such as BlueGene/L (Adiga et al., 2002), which uses a *system on a chip* to reduce energy consumption, and Green Destiny (Warren et al., 2002), which uses low-power Transmeta nodes. But their concern on energy saving is only confined to the design of hardware, with nothing to do with the strategies for power control at run-time, which also plays an important role. There is also a large effort in saving energy for desktop and mobile systems. In fact, most of the early researches in energy-aware computing were on these systems. At the system level, there has been work in trying to make the OS energy-aware by making energy the first class resource (Ellis, 1999). A number of good methods and ideas in these studies could be introduced to the energy saving schemes in cluster systems.

2.2 Job Scheduling

Migration allows tasks to be moved from their originally assigned hosts to another one, at runtime. Several virtualization software support virtual machine migration and switching off unused hosts to reduce energy consumption. Entropy(Hermenier et al., 2009), a Consolidation Manager for Clusters, is a resource manager for homogeneous clusters, which performs dynamic consolidation of resources based on constraint programming, using VM migration. (Beloglazov and Buyya, 2010) proposes a novel technique for dynamic consolidation of VMs based on adaptive utilization thresholds which reduces Service Level Agreements (SLA) violation. Also, vendors, like VMware with vSphere 4, includes a *Distributed Power Management* (DPM) that monitors virtual machines resource utilization within the cluster.

In (Srikantaiah et al., 2008) the proposed algorithm aims at finding a minimal energy allocation of workload to servers. In all these studies the objective is to minimize the energy consumption of the servers, while satisfying given performance-related bounds on the period. In (Pierson and Casanova, 2011), a model of cluster hosts that have DVFS capacities was used to calculate a bound on the optimal solution. Contrary to our work, those researches only minimize servers energy consumption while not taking into account the impact of hot spots on the cooling system. Proposed work considers the algorithm of load unbalancing to improve tasks management, taking into account the cost of migrations.

3 MODEL AND OBJECTIVES

In this section we consider a group of clusters that consists of H computing hosts, or hosts. We model each host with the unit: CPU cycles per time-unit. We also consider T services, or tasks, that run on the cluster. A task l is defined by its percentage of CPU need, l. The CPU need of a task is the CPU share it would use on a host that is dedicated to it. We will define multiple objective functions. They define optimization problems taking into account constraints that will result in efficient load unbalancing algorithms. We use an algorithm which is the opposite of load balancing because our main objective is to increase energy gain.

3.1 Model and Hypothesis

We consider an environment represented by a large-scale data center consisting of $H = \sum_{i=1}^{N} H_i$ heterogeneous physical hosts. Each cluster i has H_i hosts. There are N clusters. Each host is characterized by the CPU performance defined in Millions Instructions Per Second (MIPS). We consider T_i tasks, that run on the cluster i.

Thus, migration must take place under several constraints:

- Conservation of the execution context: It must be possible to stop the execution process of the task and restart it where it has stopped. We must be able to get the task execution context (MIPS, size, remaining size, state, memory, etc.), to transfer this state via the network, to reload and restart the task. The migration of virtual machines is used to obtain this result;
- Slowdown prevention : Job slowdown increases the execution time and therefore increases the energy consumed by hosts and impact users;
- Avoiding overloaded hosts to avoid heat points as it increases energy consumption of cooling. To measure the saturation of a cluster, we use the over-loaded threshold ε, which we call saturation. A cluster reaches saturation when its load is greater than ε.

The proposed algorithm provides solutions to these problems. The following steps are executed by the scheduler:

- Estimate the requested load (R_i) of the cluster i. This load depends on the number of task (T_i) executed by the cluster and their load ($l_{i,j,k}$ is the requested load of task k in cluster i on host j). $r_{i,j}$ is the aggregated load of all tasks on host j in cluster i.
  $R_i = \sum_{j=1}^{H_i}(r_{i,j}) \qquad r_{i,j} = \sum_{k=1}^{T_i}(l_{i,j,k})$

If task k is not running on host j in cluster i, then $l_{i,j,k}$=0.

- Computes load C_i and speed V_i of cluster i
  $C_i = \sum_{j=1}^{H_i}(c_{i,j})$
  $V_{i,j} = \sum_{j=1}^{H_i}(v_{i,j})$
  $c_{i,j}$ Actual load of host j in cluster i
  $v_{i,j}$ Maximum speed of host j in cluster i in Mips

- Job satisfaction S_i of cluster i (same for task satisfaction of host j in cluster i)
  $S_i = \frac{C_i}{R_i}$

- **constraints**
  $\forall j,k \qquad l_{i,j,k} \in [0,1]$,
  $\forall i,j \qquad r_{i,j} \in [0,S_i]$
  $\forall i \qquad S_i \in]0,1]$

- **hypotheses**

  – Communication within a cluster and between other clusters are considered as negligible;
  – For each cluster there is at least one host and one task;
  – Migration cost is considered as the same in CloudSim. To migrate a VM, only RAM has to be copied to another node. The migration time depends on the size of RAM and the available network bandwidth. M migration delay = RAM / bandwidth + C (C = 10 sec). Bandwidth is considered as constant;
  – To minimize energy consumption the load of each host tries to verify : $\forall i,j \qquad c_{i,j} \in \{0\} \cup [\gamma, 1]$
    γ is the underloaded threshold
    This equation means that each host are either switched off or with a load over γ;
  – Besides the under-load threshold we add another parameter ε corresponding to saturation and acting as a overload threshold. To verify that there is no over-loaded hosts :
    $\forall i,j \qquad c_{i,j} \leq \varepsilon$

We will use the classical linear model of power consumption in function of load :
$\forall i,j \qquad P_{i,j} = P_{min}^{i,j} + c_{i,j}(P_{max}^{i,j} - P_{min}^{i,j})$ Therefore the total power consumption of the system is: $P = \sum_{i=1}^{N}\sum_{j=1}^{H_i} P_{i,j}$ To obtain energy consumed during a time slice, instantaneous power has to be multiplied by time. Total energy is then obtained by summing

all the energy of those time slices. An objective is to minimize this energy.

3.2 Objectives

The main objective of our approach is to improve cloud's total energy efficiency by controlling cloud applications' overall energy consumption while ensuring cloud applications service level agreement. Therefore, our work must take place under several objectives :

- Ease of task Management : Managing jobs can be a tremendous task which requires many highly experienced IT experts. Providing an easily configurable system can significantly reduce the costs and ease the system management. One of our goals is to design a system which requires minimal human intervention to be configured. Moreover, once the system is deployed and configured, it becomes increasingly important to perform updates and/or add new servers. In such scenarios servers will be required to brought offline and added back later. Our goal is to design a system which is flexible enough to allow for dynamic addition and removal of servers. Finally, as system components can fail at any time, it is desirable for a system to heal in the event of failures without human intervention. Consequently, we aim at designing a system using self-healing mechanisms to enable high availability.
- Energy Efficiency: Over the past years, rising energy bills have resulted in energy efficiency to become a major design constraint for distributed systems providers. Given that traditional clouds are rarely fully utilized, significantly energy savings can be achieved during periods of low utilization by transitioning idle servers in a power saving state. However, as servers are rarely fully idle, first idle times need to be created. One of our goals is to propose task placements management algorithms which are capable of creating idle times, transitioning idle servers in a power saving state and waking them up once required (e.g. when load increases).
  - Avoiding overloaded hosts to avoid heat points as it increases energy consumption of cooling. To measure the saturation of a cluster, we use the over-loaded threshold ε, which we call saturation. A cluster reaches saturation when its load is greater than ε.

4 ENERGY AWARE CLOUDS SCHEDULING USING ANTI-LOAD BALANCING ALGORITHM (EACAB)

Selection policy selects appropriate tasks for migration. Location policy determines then suitable hosts to receive them. In other words, they locate complementary hosts to/from which they can send/receive tasks. Current version of our algorithm uses tasks load to take those decisions. In the following, we present an algorithm (EACAB) based on the merged principle of Comet(Jeon et al., 2010) and of anti-load balancing.

4.1 Credit based Anti-load Balancing Model

The algorithm proposed in this article aims at maximizing *Credit* which is a value used when calculating the energy-efficiency of the system behavior.

This Credits algorithm is an adaptation of the *The Comet Algorithm* (Chow and Kwok, 2002), a load balancing algorithm. Comet is based calculating credit for mobile agent. Each agent in Comet is trying to maximize its own credit by moving between hosts. An agent a_i uses the following formula:

$$C_i = -x_1 w_i + x_2 h_i - x_3 g_i$$

Where w_i : computation load of the host running agent a_i, h_i and g_i : communication load inside and outside agent a_i, and where x_1, x_2 and x_3 are positive float coefficients which constitute dependence assigned to each agent from its creation to estimate its affinity relative to their host. Thus an agent will move to a new host if it result in a lower host load, or if it reduces external communication or if it increases internal communication. This algorithm does not take int account migration cost.

In the same way, the proposed algorithm in this article works by associating a credit value with each host. The credit of a host depends on the host, its current workload, its communications behavior and history of task execution. When a host is under-loaded (load $<$ globally_defined_threshold), all its tasks are migrated to a comparatively more loaded host.

In dynamic load unbalancing schemes, the two most important policies are *selection policy* and *location policy*. Selection policy concerns the choice of the host to unload. Location policy chose the destination host of these moved tasks. An important characteristic of selection policy is to prevent the destination host to become overloaded. Also, migration costs must be compensated by the performance improvement.

Each host has its own *Credit*, which is a float value. The higher a host Credits, the higher its chance its tasks to stay at the same host. The credit of a host increases if:

- Its workload or the number of tasks in the host increases;
- Communication between its tasks and other hosts increases;
- Its load increases while staying between the under-loaded threshold γ and the over-loaded threshold ε.

On the contrary, the credit of a host decreases in the cases below:

- Its workload or its number of tasks decrease;
- It has just sent or received a message from the scheduler which indicates that the host will probably become empty in a short while.

The Credit of a host will be used in the selection policy: the host which credit is the lower is selected for tasks migration. The location policy identifies the remote host with the highest credit which is able to receive the tasks selected by the selection policy without being over-loaded. Figure 1 shows an example of migration. The percentage represents here the occupancy rate of each task on the host.

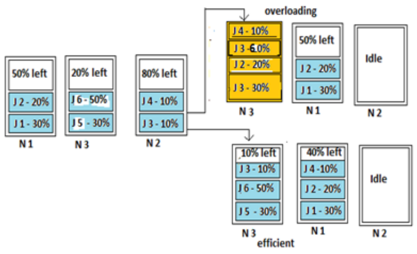

Figure 1: Different contexts for a migration.

4.2 Algorithm Description

In Comet mobile agents move between hosts according to their affinities (credit) to achieve load balancing. Here we work with tasks which migrate depending on the load of the host. We apply the Credit concept to the migration of tasks. EACAB algorithm is based on the technique of calculating credit ($\sigma_{i,j}$) of each host (j in cluster i) by the same method of Comet(Jeon et al., 2010). In EACAB, the formula is then:

$$\sigma_{i,j} = c_{i,j} - r_{i,j}t_{i,j} + \varepsilon - \gamma$$

Were $c_{i,j}$ is the actual load of host j in cluster i, $r_{i,j}$ is its requested computation load, $t_{i,j}$ is its task satisfaction, and γ and ε are respectively under-load and over-load threshold.

EACAB provides task scheduling strategy, which dynamically migrate tasks among computing hosts, transferring tasks from underloaded hostssec4 to loaded but not overloaded hosts. It balances load of computing hosts as far as possible in order to reduce program running time.

The decision making algorithm behaves globally as follows:

- If $\sigma_{i,j} < 0$, the host j of cluster i is over-loaded or under-loaded.
- If $c_{i,j} > \varepsilon$, the host j of cluster i is over-loaded
- If $c_{i,j} < \gamma$, the host j of cluster i is under-loaded

This algorithm is described in Algorithm 1. For the sake of simplicity, corner cases such as all nodes over-loaded are not included. Selection policies take into account credits and migration cost. The selected host (node j' in cluster i') is the one with the minimun $\sigma_{i',j'}$ weighed by the migration cost between the current position of the job and the potential host. If $\tau_{i,j,i',j'}$ is the migration cost between the node j in cluster i and the node j' in cluster i', the selected host is the one that minimize : $\sigma_{i',j'} \cdot \frac{\tau_{i,j,i',j'}}{Max_{i'',j''}(\tau_{i,j,i'',j''})}$

4.3 Analysis of the Algorithm

This migration algorithm's goal is to minimize the energy. It is composed of two parts. In the first part, it checks for each host i if the load is below the threshold. If this is the case, it locates the host j that will receive all tasks of host i. The second part manages hotspots. To reduce the load of an overloaded host, it begins to migrate the slowest task. Selection policy will choose the task that will stay the longest on the host. Policy of localization will then identify the host that will receive the task without exceeding its capacities (ie. its load after migration will still be under ε). So this host will be the new destination of the task.

5 EXPERIMENTS AND RESULTS

In order to evaluate the gains of EACAB compared to classical algorithms, we implemented this algorithm in EnerSim. This simulator is based on GridSim(Calheiros et al., 2011), where we added power consumption (extended from CloudSim(Buyya and Murshed, 2002) implementation) and virtual machine (mainly their migration). It is a java event driven simulator of grids, cluster and Clouds. It provides information about execution times, but also about instan-

Algorithm 1: Energy aware clouds scheduling using anti-load balancing algorithm (EACAB).

```
Calculate $c_{i,j}$ , $\sigma_{i,j}$     // Load, credit of node j in cluster i
Sort in ascending order hosts according to the value of their credit
for ($M_{orig}$ in sorted host $j$ in all cluster $i$) do
  Update $c_{i,j}$ , $\sigma_{i,j}$ for remaining hosts
  Sort the remaining hosts (per load)
  if ($c_{i,j} < \gamma$) then
    // In case $M_{orig}$ is under-loaded
    for ($M_{dest}$ in all host $j'$ in all cluster $i'$) do
      if $M_{orig} \neq M_{dest}$ and ($c_{i',j'} \geq \gamma$) and ($c_{i,j} + c_{i',j'} < \varepsilon$) then
        Add $M_{dest}$ to potential destination set $Potential$
      end if
      Migrate all task from $M_{orig}$ to the element with lower credits weighted by migration cost in $Potential$
    end for
  else
    // $M_{orig}$ can be over-loaded
    while ($c_{i,j} > \varepsilon$) do
      // In case $M_{orig}$ is over-loaded
      Calculate $l_{i,j}^{min}$   // load of the lightest task in $M_{orig}$
      for ($M_{dest}$ in all host $j'$ in all cluster $i'$ sorted by their credits) do
        if (($M_{orig} \neq M_{dest}$) and ($c_{i',j'} + l_{i,j}^{min}$ ¡ $\varepsilon$) then
          Migrate task from $M_{orig}$ to $M_{dest}$.
        end if
      end for
    end while
  end if
end for
```

taneous power consumption and energy consumption of tasks.

5.1 Simulation Environment

- Grid : 100 clusters of 100 hosts each. Each host speed is between 1GHz and 3.06GHz
- Jobs : 1000 randomly generated tasks
  - Duration between 10 and 40s
  - Requested load between 10% and 100%
- Host shutdown and wakeup energy are assumed to be zero as they are fast compared to the execution time of tasks.
- Hosts have two different power states for each core: Switched on and switched off. While switched on, power consumption is linear in function of load between P_{min} and P_{max}. Those values are different for each host and are respectively between 75 and 150W, and 200 and 250W.

In the following we compare EACAB with other algorithms.

- Dynamic First Fit: a dynamic *First Fit* where host are sorted according to their maximum power consumption.

5.2 Experimental Results

It is designed to be a centralized coud scheduler that emphasizes on cloud scheduler interoperation, and complemented by a dynamic resource discovery approach on centralized network.

In this subsection, we describe the simulation study performed to evaluate the performance of our algorithms in terms of energy minimization as well as the execution time and the number of migrations.

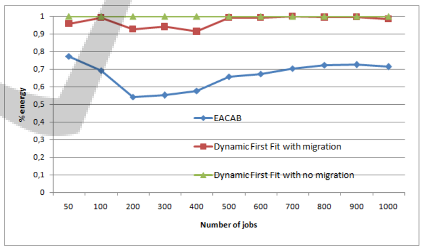

Figure 2: Energy of algorithms compared to dynamic first fit with unsorted hosts. Lower is better.

5.3 Algorithm Description

The first observation is that for two algorithms, EACAB consumes the least energy while Dynamic First Fit algorithm consumes the most energy (see figure 2), when the number of jobs $T > 350$. For a small number of tasks our algorithm leads to a significant energy consumption. Our EACAB algorithm performs even better. The second observation is that EACAB algorithm is able to reduce the energy consumption by 5 percent to 20 percent when job increases from 350 to 1000.

Figures 3 and 4 show respectively the maximum and median number of switched on host as a function of task number. When jobs increase then the number of nodes switched on also increase, leading to a higher power consumption. This is particularly true if there is no migration after the initial placement of tasks. Hence, the gain of our algorithm increases

Table 1: Makespane of algorithms compared to First Fit with sorted hosts.

	Number of Jobs									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Dynamic first Fit	265.53	374.4	580.37	631.22	615.04	870.55	813.83	1006.31	853.68	1270.73
EACAB	230	225.81	321.95	399.45	495.09	589.84	565.48	872.3	738.43	815.65

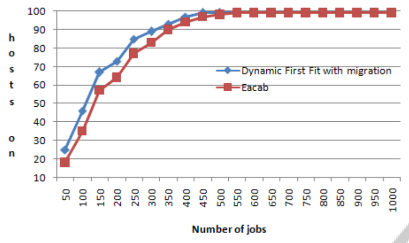

Figure 3: Maximum host switched on with EACAB.

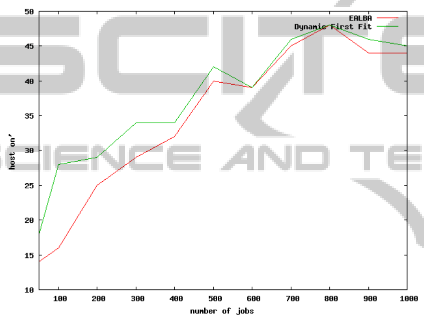

Figure 4: Median host switched on with EACAB.

power-wise with the number of tasks because migration is activated. Computing resources are fully used in both cases at the start of the experiment. In the case of the consolidation, less hosts are switched-on because we can adapt to the workload dynamism. The median has the same behavior but the maximum number of hosts is 50%. The observed gain increases with the number of tasks and becomes constant when hosts are saturated.

The good results of EACAB comes from the fact that with the increase of the number of tasks, it has more possibilities to migrate tasks. It can then better allocate tasks on computing resources, reducing the number of switched-on hosts.

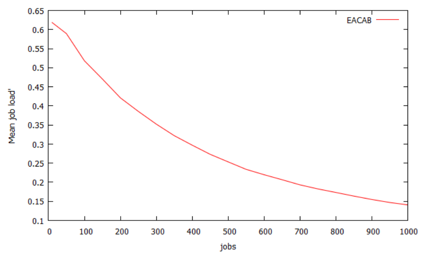

Figure 5: Job mean load with EACAB.

Due to the thresholds of EACAB, it would be possible to reduce further the number of switched on hosts but it would overload remaining hosts. Those hosts would become hot points and would have a negative impact on cooling. In order to prevent overloading, EACAB adjusts load as shown in Figure 5. If the number of tasks increases, it will reduce the mean actual load they will obtain.

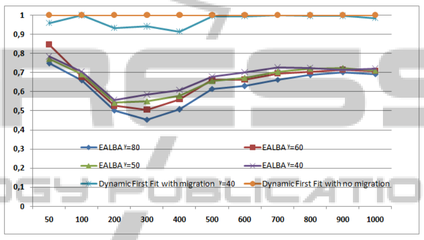

Figure 6: Variation of threshold vs energy gain with EACAB

Figure 6 shows that our algorithm is better than classical *Dynamic First Fit* regardless of the threshold when the number of tasks is important.

The choice of γ is still important. There is a energy consumption difference of 10% between the best and the worst value. The worst value is 10% more efficient than dynamic first fit, the best one is 20% more efficient.

For small number of tasks, energy consumption increases because of the many migrations. To choose this value it is omportant to consider the dynamisme of the tasks. As said previously, increasing γ, reduce energy consumption at the cost of consolidating more and more the tasks.

EACAB has the shortest execution time when the number of jobs increases. The result implicates that the scheduling algorithm such as EACAB can leverage interconnects with migrations to achieve high performance and energy efficiency. Table 1 shows that our algorithm produces faster scheduling regardless of the number of jobs.

6 CONCLUSION

In this paper, we presented and evaluated our energy-efficient migration algorithm for clouds. This algorithm is based on the principle of *Anti-load balancing*.

It provides energy-efficiency improvement compared to classical load unbalancing algorithms.

Our main problem was to optimize energy consumption given task performance constraints. Energy consumption is to be taken in a broad way as we try to prevent hot spots to reduce impact on cooling.

We have compared EACAB to classical solutions over a range of problem instances using simulation. EACAB parameters lead to a family of heuristics that perform well in terms of energy savings while still leading to good task performance.

EACAB consolidate tasks on a subset of the cluster hosts judiciously chosen depending on the characteristics and state of resources.

This algorithm has a low computational cost. It can then be employed in practical settings. Overall, the proposed EACAB algorithm can compute allocations effectively with an important energy gain. Experiments showed that with our algorithm we obtained a 20% gain over standard algorithms. However, it is important to investigate further how to improve the quality of service, but also the optimization algorithm.

Also current version of EACAB is centralized. We aim at distributing this algorithm, so that each cluster can exchange tasks, based on their respective credits. Future version of EACAB will also take int account other measures to compute Credit such as network communication patterns.

REFERENCES

Adiga, N. R., Almási, G., Almasi, G. S., Aridor, Y., Barik, R., Beece, D., Bellofatto, R., Bhanot, G., Bickford, R., Blumrich, M., et al. (2002). An overview of the bluegene/l supercomputer. In *Supercomputing, ACM/IEEE 2002 Conference*, pages 60–60. IEEE.

Beloglazov, A. and Buyya, R. (2010). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, page 4. ACM.

Buyya, R. and Murshed, M. (2002). Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. volume 14, pages 1175–1220. Wiley Online Library.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. volume 41, pages 23–50. Wiley Online Library.

Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. (2001). Managing energy and server resources in hosting centers. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 103–116. ACM.

Chow, K.-P. and Kwok, Y.-K. (2002). On load balancing for distributed multiagent computing. *Parallel and Distributed Systems, IEEE Transactions on*, 13(8):787–801.

Ellis, C. S. (1999). The case for higher-level power management. In *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*, pages 162–167. IEEE.

Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G., and Lawall, J. (2009). Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 41–50. ACM.

Jeon, H., Lee, W. H., and Chung, S. W. (2010). Load unbalancing strategy for multicore embedded processors. volume 59, pages 1434–1440. IEEE.

Lawson, B. and Smirni, E. (2005). Power-aware resource allocation in high-end systems via online simulation. In *Proceedings of the 19th annual international conference on Supercomputing*, ICS '05, pages 229–238, New York, NY, USA. ACM.

Pierson, J.-M. and Casanova, H. (2011). On the utility of dvfs for power-aware job placement in clusters. In *Euro-Par 2011 Parallel Processing*, pages 255–266. Springer.

Pinheiro, E., Bianchini, R., Carrera, E. V., and Heath, T. (2001). Load balancing and unbalancing for power and performance in cluster-based systems. In *Workshop on compilers and operating systems for low power*, volume 180, pages 182–195. Barcelona, Spain.

Srikantaiah, S., Kansal, A., and Zhao, F. (2008). Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10. USENIX Association.

Thiam, C. and Da Costa, G. (2011). Anti-Load Balancing to Reduce Energy Consumption (student paper). In Ivnyi, P. and Topping, B. H. V., editors, *International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Ajaccio-Corsica-France, 13/01/2011-15/04/2011*, page (on line), http://www.civil-comp.com/conf/progp2011.htm. Civil-Comp Proceedings.

Warren, M., Weigle, E., and Feng, W. (2002). High-density computing: A 240-node beowulf in one cubic meter. In *Supercomputing 2002*.