Thermal-aware cloud middleware to reduce cooling needs

Violaine Villebonnet, Georges da Costa

Thermal-aware cloud middleware to reduce cooling needs

Violaine Villebonnet
IRIT
Université de Toulouse
violaine.villebonnet@irit.fr

Georges Da Costa
IRIT
Université de Toulouse
georges.da-costa@irit.fr

***Abstract*—As we are living in a data-driven world and directed toward internet, the need for datacenter is growing. The main limitation for building cloud infrastructures is their energy consumption. Moreover, their conception is not perfect because servers are not the ones consuming all the power, cooling systems are responsible for half of the consumption. Cooling costs can be reduced by intelligent scheduling, and in our case through virtual machines migrations. In this paper, we propose a dynamic reconfiguration based on evolution of temperatures and load of the servers. The idea is to share heat production to reduce cooling costs and consolidate the workload when possible to reduce servers costs. The challenge resides in satisfying these opposite objectives. We tested our algorithm on an experimental testbed, and achieve to cap the temperature of the datacenter room while not forgetting to optimize the server use, and without impacting on applications performance.**

***Keywords*—Thermal-aware, cooling, scheduling, cloud middleware, energy efficiency**

I. Introduction

Global electricity consumption of cloud computing, compared to total energy consumption of individual country, would be at the fifth rank just before India according to Greenpeace's report of 2012 [1]. And inside a datacenter, the whole electric power does not go toward computing: cooling systems consumption is up to half of the total power consumption. The effective power consumed by the servers is only 36%, while the rest is due to energy provisioning and losses [2]. Electricity consumed by servers produces heat they cannot handle. Indeed, high temperature causes hardware degradation [3] and is a main factor which reduces MTBF (Mean Time Between Failures).

Datacenters are usually cooled by cooling units and fans, whose functioning are controlled by the changes of temperature in the room. A slight increase of the ambient temperature, even just at a local spot of the room, implies a significant increase of the cooling system power needed to maintain a reasonable global temperature. Therefore heat produced by servers is an important issue and should be carefully monitored. By controlling and reducing each server temperature, we consequently reduce the room temperature.

Considering the whole datacenter, energy savings can not be achieved only by optimizing the energy efficiency of the servers. A trade-off between reducing cooling systems costs with optimizing the use of servers leads to the best global efficiency.

We address this issue by dynamically placing workload, through migrations of virtual machines, according to measures from temperature and load sensors located on each node. Temperature of a server is closely linked to its CPU load but its evolution is different in terms of range and time: load variation is instantaneous while heat takes some time to appear. Moreover, other parameters influence server temperature such as ambient temperature of the room and server location in the data center due to heat transferred by nearby servers. Because our goal is to reduce cooling costs, it is not enough to only consider CPU load. Our algorithm deals with real time data like temperature that affects directly cooling systems.

After an overview of the state of the art in the next section, we describe the algorithm in section III. Implementation of the algorithm in a cloud manager is detailed in section IV, as well as its deployment on the testbed platform Grid'5000, and the obtained results. Section V raises some possible perspectives and section VI concludes the work.

II. State of the art

Prior works on reducing datacenter power consumption can be classified into two trends : the ones whose goal is to reduce servers power consumption, and the others which are concerned by the consumption of the cooling infrastructure.

Reductions of servers energy consumption are accomplished by consolidation of the workload on the smallest possible number of nodes. The goal is to free the lowest loaded nodes in order to perform energy savings actions like shutdown or suspend. This way when a server is powered on, its static cost is shared between several virtual machines. Different approaches for consolidation have been proposed : Heuristics inspired by the bin-packing problem and variants of the best fit algorithm [4] [5], ant colony based approach [6], decentralized algorithms through gossiping which do not need a superior entity to take migration decisions [7], and Entropy which relies on a dynamic constraint satisfaction problem [8]. But these works only focus on optimizing the energy efficiency of the servers, without considering the thermal impacts. Indeed, scheduling decisions have a huge impact on how the cooling systems behave and thus how much power they consume.

Concerning datacenter thermal management, some work focus on the layout of the servers in order to optimize the flow of hot and cold air and especially to minimize the hot air recirculation [9]. Organization of the racks alternating cold and hot aisles [10] is now a common use. The CoolEmAll

project [11] studies in details all the parameters affecting the energy efficiency, by deep simulations of a datacenter and all its characteristics: hardware configurations, applications types, scheduling policies and cooling system settings.

Besides, some people try to reduce cooling costs in more exotic ways. Follow-the-moon approach [12] works with companies having datacenters in different locations. The idea is to always transfer the work towards the datacenter where it is currently night time. They take advantage of the cheaper electricity cost, but on top of that, temperatures are also lower at night. Recently, more datacenter operators are opting for cold climates and Nordic countries to settle down their servers like Facebook in Sweden or Google in Finland.

More recent work tries to optimize both servers and cooling consumption at the same time [13]. In fact they split the datacenter in three different zones according to the average temperature : cold, warm and hot. They perform consolidation in the cold zone, while in the warm zone they try to share the workload, and the hot zone is only used in case other zones are already fully loaded.

These previous works are only conducted and validated over simulations, and consequently can not model perfectly the complex behaviour of the system. Contrarily, the following works benefit from real testbeds. In [14], authors propose to place the jobs in priority to the most efficiently cooled servers. They determined the different regions of efficiency through measurements done in their experimental testbed equipped with several temperature sensors. Unfortunately, their job placement is only static, and do not take into account evolution of load or temperature. In [15], authors focus more on the heat generation and extraction at different points of a datacenter, resulting in a metric called heat imbalance, and use this model to decide where to place the workload. Detailed measurements from the datacenter and knowledge about the spacial configuration is needed to build these models and metrics. In contrast, our reconfiguration is purely autonomic, do not need knowledge of the datacenter topology because it is only based on current evolution of the system. We aim at reducing the cooling costs only by the simple assumption to limit the maximum temperature for any server at any time.

III. Contributions

A. Objectives

Our work aims at different objectives :

1) Avoiding hot spots: Hot spots, meaning peak of temperature on some isolated elements, are really inefficient because they force the air conditioning units to cool the entire datacenter. As pictured in Fig.1, A/C units only have a global impact on the room through the cold air they provide. Their functioning only relies on the incoming air temperature and on maximum temperature monitored on each node, and even if the latter increases by only 1 or 2 degrees, it implies a power peak consumption for the A/C to make a bigger effort on cooling the whole datacenter.

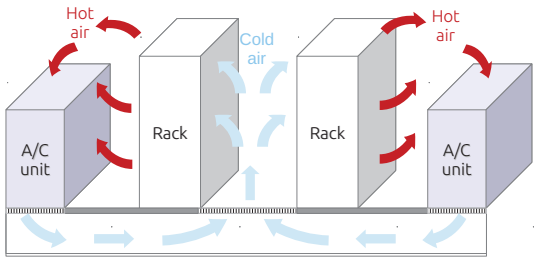

Fig. 1. Architecture of a datacenter with cold/hot aisle.

2) Optimize servers power consumption: Because power proportional hardware are not yet available, we have to care about servers fix power costs. Indeed, when a server is powered-on but idle, it already consumes half of its consumption when fully loaded [16]. So we have to make sure that utilization of servers is optimal, that is to minimize the number of used servers in order to shutdown the idle ones.

B. Reconfiguration algorithm

Now we present the two concepts we use to reach the two objectives:

1) Cap the temperature: To avoid hot spots, we decided to put the priority on not crossing a maximum temperature threshold for all servers at any time. This way, temperature peaks disappear, and the cooling costs remain low and stable because the cooling effort is constant.

Fig.2 is an illustration of the impact of virtual machine placement on temperatures. On the left, the initial situation where machine 1 is fully loaded with 2 virtual machines, and machine 2 loaded at 20% hosting 1 virtual machine. M1 is a hot spot, its temperature of $60\,^{\circ}\mathrm{C}$ is too high, that is why a decision to migrate VM2 towards M2 is made. On the right is shown the situation after the migration: the workload is more equally shared and this results in an uniform temperature of both machines around $50\,^{\circ}\mathrm{C}$

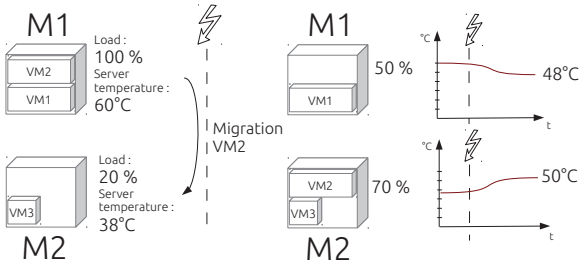

Fig. 2. Example of the principle impact of migrating a virtual machine on server heat.

In our algorithm, we use the concept of anomaly. For avoiding hot spots, we defined the Overheat anomaly which is triggered when a node temperature crosses the maximum fixed threshold. In this case, we want to reduce the load of the

node in order for it to cool down. The solution is to migrate the virtual machines hosted by this node towards another one, but without impacting too much on the temperature of the destination. Indeed it would be very inefficient to resolve an overheat anomaly by overheating another server. That is why the destination for the migration is chosen as the coldest node of the cluster.

If this is the only action we do in the system, it will then result in a case where the workload is uniformly shared between the nodes. When the load is stable, then it is the best configuration in terms of cooling costs because the heat production is uniformly distributed. But if the load decreases, servers will be poorly used, and it is the least configuration in terms of servers costs because all machines are powered on for only little load, which is very energy inefficient.

2) Consolidate the workload: To optimize servers utilization, we defined another anomaly we refer to as UNDERLOAD. It corresponds to a node loaded under a certain threshold. The solution to solve this problem is to consolidate the virtual machines on a few number of hosts. This action brings significant energy savings because all the newly released nodes can be shutdown.

The minimum load threshold should be fixed accordingly to the static consumption of the servers. Higher is the idle consumption of the server, higher should be the minimum load threshold. We detail the impact of the thresholds in the Experiments section IV.

Consolidation may seem to be an easy action to undertake, but in the case of our algorithm, we need to carefully consider what will be the impact of the consolidation on the temperatures. Indeed, consolidation should not be too brutal or it would rapidly produce an overheat anomaly on the destination host. This overheat would then be detected by the algorithm, and the action to solve this problem would be to undo this previous consolidation.

C. Find good balance between two opposite objectives

Our algorithm can be summarized in two actions that are actually equivalent to two opposite movements: sharing and consolidating the load. Obviously, this implies they must be performed wisely to avoid a chaotic behaviour. Therefore we introduce a priority: overheat anomalies must be handled first. As a matter of fact, overheat are resolved by sharing the load, so this can clear up the eventual underload anomalies. Moreover, our main concern here is to avoid hot spots, so we want to deal with overheat as quickly as possible.

Of course, both thresholds should be chosen in agreement with each other, otherwise our algorithm has no chance to achieve a satisfying result.

IV. EXPERIMENTS

A. Adding thermal-aware feature to Snooze

We chose the cloud manager Snooze [17] to implement and test our algorithm. It was developed by Eugen Feller during his PhD at Inria Rennes. The main particularity is its hierarchical architecture that allows it to scale across a large number of servers and virtual machines, and to have features like self-configuring and self-healing in case of failures.

In order to test our algorithm in Snooze, our first task was to add the temperature monitoring feature to the software. All servers are equipped with hardware temperature sensors to prevent themselves from critical overheat. To exploit these sensors, we use functions from the Linux *lm-sensors* package that can read temperature information from the hardware.

The distributed monitoring system Ganglia [18] has been integrated into Snooze to retrieve monitoring information periodically. Ganglia is well suited for the hierarchical architecture of Snooze and has the same characteristics of scalability and robustness. Monitoring data is gathered every second by each node of the cluster, and send to the supervisor node. This node stores locally the data with the aim of using it later to take reconfiguration decisions. Ganglia can monitor several measures at the same time, and is completely extensible by adding custom modules. Indeed, to get temperatures, we have written a module in Python which interacts with the sensors through *lm-sensors* functions.

B. Implementation of the algorithm

A periodic threshold crossing detection is performed on each host every five seconds. If a crossing is actually detected, an anomaly is raised. The detection of the crossing is performed using the average of the five last values as measures are done every second. The anomaly type is set according to which threshold has been crossed. The node called Group Manager, which supervises the hosts, which are called Local Controllers in Snooze language, is aware of the problem by receiving the anomaly notification. This hierarchically superior node is the one responsible of making decisions in order to resolve anomaly, and triggers according actions.

On Fig.3, we can see the order of the successive actions.

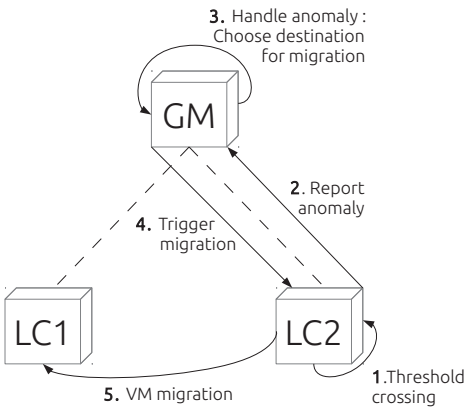

Fig. 3. Snooze hierarchy decisions. GM (Group Manager) node manages several LC (Local Controler). Each server hosts one LC.

C. Grid'5000 testbed

Experiments were carried on the French testbed Grid'5000 and more precisely on the cluster named parapluie in the

Rennes site. Servers of this cluster are equipped with AMD Opteron 6164 HE processor, which has 12 cores running at 1.7GHz. These nodes are powered by power distribution units (PDU) which export power monitoring information using the Simple Network Management Protocol (SNMP). We get power consumption data every second for each node that are part of our experiments, and we are able to compute the total energy consumed at the end of the run. This system allows us to know the impact of our algorithm on the energy consumption of the servers.

The room that hosts these nodes is cooled by air conditioning units, which are not yet monitored. Moreover, as this room hosts other servers of the Grid'5000 testbed, the ambient temperature, and thus the power consumption of the A/C depends on the other jobs currently running in the whole room. Each server has its own fan to cool it down when its temperature is too high. These fans are part of the cooling system, but their power consumption is included in servers power consumption. Next section and graph show that actions of the server fans are noticeable.

D. Measures on a single node

Fig.4 shows the evolution of CPU load, temperature and power consumption of a single node. It pictures the close relationship between these three metrics. CPU load starts from 0%, stays at 100% during 10 minutes and then decreases to 0% for another 3 minutes.

We can see that the temperature evolves much slower than load and power consumption. Starting from approximately 38 °C, it reaches its peak value, almost 48 °C after 3 minutes of full load. When the load stops, the temperature slowly decreases to its original value, taking also 3 minutes. We can noticed fluctuations in the temperature during the 10 minutes of load, these are due to actions from the cooling system, meaning the server fan. Furthermore, these actions are also visible on power consumption graph as we can observe slight peaks of consumption, in link with the slight decreases of temperature. The power consumption itself varies from 170W when idle and around 220W when fully loaded.

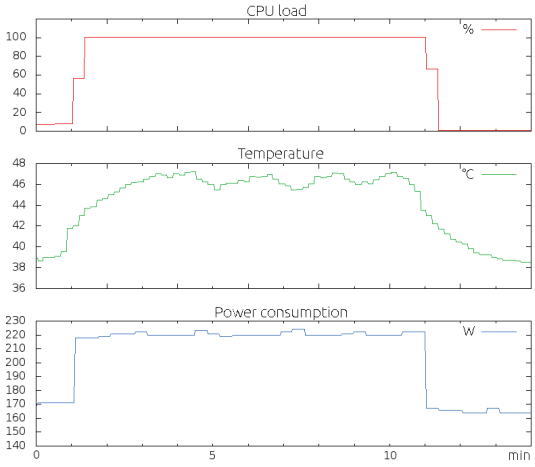

Fig. 4. Evolution of load, temperature and power consumption on one host for a simple ON/OFF workload.

E. Experimental validation

The results presented below are all obtained with the same following configuration. Three physical machines are used as hosting nodes, and six virtual machines are created. Each computing node has 12 cores and each virtual machine is set up with 2 virtual CPUs. In this configuration, one computing node would be sufficient to host all the virtual machines. If so, the hosting node would be a hot spot, and we want to show that our algorithm brings a better trade-off in term of cooling. In fact it would be even more relevant if the total load is a little more than 100% because instead of having one host fully loaded and an other one just powered on for little load and then creating a hot spot, we propose to share the load and the heat. Experiments will illustrate our algorithm using CPU intensive applications. The virtual machines are loaded using the cpuburn application, with one instance of the application for each virtual CPU.

When the load stays high, only the temperature threshold is used. In this case, all the hosts are used and then the only goal is to manage the maximum temperature, by sharing the virtual machines. The following Fig.5 shows an example of such a situation. We can see that all virtual machines are firstly launched on Host A, which shortly causes an overheat. To solve this, all the virtual machines are migrated to other nodes, whose temperature hits the maximum threshold after few minutes. The virtual machines are then migrated back to the previous node, which has just cooled down during the last minutes. This same ping-pong pattern is repeating over the rest of the experiment. These successive migrations shows that for this configuration it is not possible to reach a stable state while staying under this maximum temperature threshold.

As migrations are not the same timescale as power impact on heat, it is possible to spread heat production between servers in order to prevent host spots. As a reminder, in those experiments live-migration is used and thus there is not much impact on the applications running inside the virtual machines. Indeed, migrations introduce a small increase to the total execution time of applications. If for certain reasons, migration cost is more important than the one in our experimental platform, then threshold should be adapted accordingly.

At the opposite of static placement whose goal is to find the best configuration at first try and never change it again, here we take into account the system evolution to always try to keep the best configuration. Contrary to classical scheduling, in our case having a periodic behaviour can be positive as it helps levelling the servers temperature. For our cooling concern, it is not possible to ignore the behaviour of the system because scheduling has an impact on temperatures, that we want to control. Moreover, as we saw previously, temperatures evolution is not linear and can be influenced by factors we cannot precisely predict.

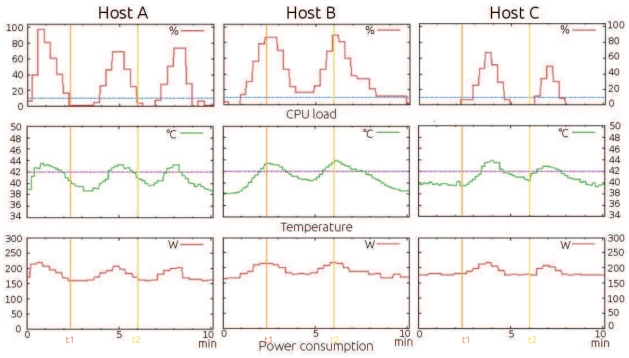

Fig. 5. 6 VMs using each 2 virtual CPUs start on Host A at t=0min. They are migrated to prevent OVERHEAT anomaly.

Fig.6 shows an example where a stable state is reached. The beginning of the experiments is the same, all virtual machines are launched on Host A which becomes overheated. Virtual machines are then shared between Host B and C. This action produces a small overheat on both hosts, just enough for them to migrate one virtual machine each back to Host A. So at approximately minute 4, each of the three nodes hosts two virtual machines, and their temperature are stabilized just under the threshold of $44\,^{\circ}\mathrm{C}$. This is typically a case where the placement is now equivalent to the Round Robin policy. But when the load drops under the minimum CPU load threshold, an action of consolidation should be taken in order to optimize the energy efficiency. The remaining idle hosts after this consolidation could then be shut down, or put in a low power state in order to save more energy. In the case where 4 hosts would be present, this approach would be equivalent to a mix of consolidation on only 3 hosts with a Round Robin algorithm on these remaining hosts.

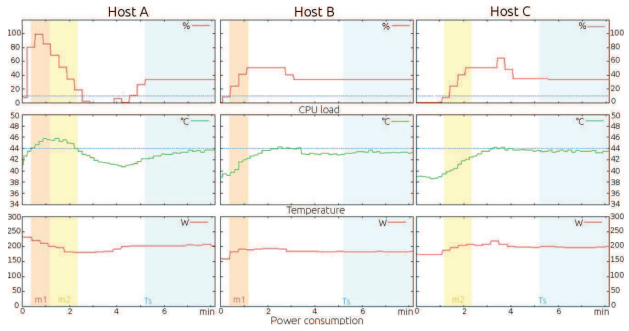

Fig. 6. Stable state reached at time Ts as the OVERHEAT anomaly threshold is at $44\,^{\circ}\mathrm{C}$ compared to $42\,^{\circ}\mathrm{C}$ in Fig. 5. m1 corresponds to migrations from Host A to B and m2 to migrations from Host A to C

As our algorithm is based on a threshold crossing principle, we have run the experiments while varying the threshold values to evaluate the impact on energy consumption of servers. Fig.7 is composed of two graphs which show respectively the number of migrations and the energy consumption at five different temperature thresholds. The scenario is the same as previously described with 3 hosts and 6 virtual machines. They are fully loaded during 10 minutes and then idle for 5 minutes.

Higher the temperature threshold is, less the number of migrations is. When the allowed temperature rises, each node can accommodate some load for a longer time and thus reduce the number of needed migrations. Meanwhile, higher the maximum temperature is, less the energy consumption is. Here the energy consumption correspond to the three hosting servers without including the cooling infrastructure. There is an outstanding value for energy consumption at threshold $42\,^{\circ}\mathrm{C}$ that we cannot explain but we are doing some more experiments to see if it is recurrent.

When the temperature threshold is low, it implies that the load must be shared between more nodes. In this case, the power consumed by the server part is bigger because more of them are powered on, and this involves higher fix costs. But what is relevant when the temperature is lower, is that the cooling system is less needed because the heat produced is uniformly distributed. So the cooling costs, which are not shown here, are reduced.

On the other hand, when the temperature threshold is high, the load can be consolidated on a smaller number of hosts, so the server consumption part is less important because the remaining idle servers can be shut down. But this means there are more hot spots and implicates an increase of the cooling consumption. In this case, cooling systems have to make a huge effort to cool these hot spots, which means to cool all the room even for the rest of the servers which do not need it.

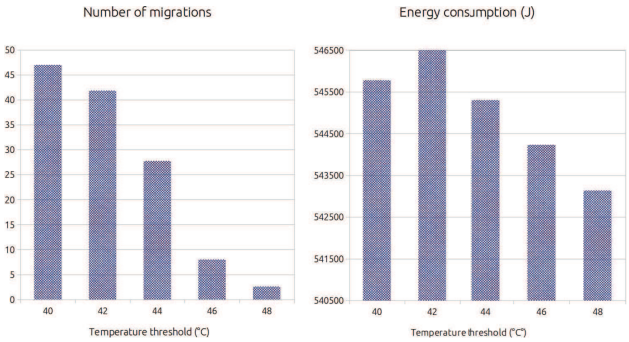

Fig. 7. Impact of the temperature threshold on number of VM migration and on server energy consumption.

V. PERSPECTIVES

As our algorithm is based on thresholds, it should be interesting to have thresholds that can adapt themselves according to different parameters. For instance, for the temperature one, we can imagine a threshold self-adapting to the outside temperature, the current overall load in the datacenter, or any other meaningful criterion.

To improve the decisions made for migrations, and especially to choose the better suited host as destination, a predictive temperature model should be included to our work. The current coldest node might not be the coldest node in one or two minutes if it has just received some new load. And this kind of recent events can not be taken into account in our current approach of taking the average of the last temperature values.

In the same idea, time management should be improved. The goal would be to take into account impact of recent events on the system before taking any other decisions. For instance, after a migration the temperature is not decreasing instantly on the source host. The algorithm should wait for the temperature to stabilize before performing further migrations. This would bring more stability, meaning reduce the ping-pong migrations pattern, and avoid making thoughtless decisions.

VI. Conclusion

Our algorithm deals with temperature at the level of the data center by imposing a maximum threshold to all the servers. Because the cooling system only have a general view of what is going on in the room from its overall temperature, we want to reduce the temperature in a global way. Our approach tends to increase the power consumption of the servers, to reduce power consumption of the cooling infrastructure. It eliminates hot spots, which are responsible for high cooling costs.

One of the main problem in datacenters is that servers cannot handle high temperatures even though they have to produce large amount of heat to achieve their computing goal. Managing their heat is the key to reduce overall datacenter power consumption while avoiding potential hardware failures.

Acknowledgments

The authors would like to thank Grid'5000 staff, and the people from Inria Rennes Myriads team, especially Matthieu Simonin for his precious help with Snooze.

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] Greenpeace International. How clean is your cloud ?, 2012.

[2] Richard Sawyer. Calculating total power requirements for data centers. APC White Paper #3, 2001.

[3] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz Andre Barroso. Failure trends in a large disk drive population. In *5th USENIX Conference on File and Storage Technologies*, 2007.

[4] Damien Borgetto, Henri Casanova, Georges Da Costa, and Jean-Marc Pierson. Energy-aware service allocation. *Future Generation Computer Systems*, 2012.

[5] Anton Beloglazov and Rajkumar Buyya. Energy efficient allocation of virtual machines in cloud data centers. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010.

[6] Eugen Feller, Louis Rilling, and Christine Morin. Energy-aware ant colony based workload placement in clouds. In *12th IEEE/ACM International Conference on Grid Computing*, 2011.

[7] Moreno Marzolla, Ozalp Babaoglu, and Fabio Panzieri. Server consolidation in clouds through gossiping. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2011.

[8] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: A consolidation manager for clusters. In *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, 2009.

[9] Q. Tang, Sandeep K. S. Gupta, and G. Varsamopoulos. Thermal-aware task scheduling for data centers through minimizing heat recirculation. In *IEEE International Conference on Cluster Computing*, 2007.

[10] Robert F. Sullivan. Alternating cold and hot aisles provides more reliable cooling for server farms. White Paper, Uptime Institute, 2000.

[11] Micha Berge, Georges Da Costa, Andreas Kopecki, Ariel Oleksiak, Jean-Marc Pierson, Tomasz Piontek, Eugen Volk, and Stefan Wesner. Modeling and simulation of data center energy-efficiency in coolemall. In *Energy Efficient Data Centers*, Lecture Notes in Computer Science. 2012.

[12] Jeff Klaus. Follow-the-moon scheduling to lower energy costs. http://www.datacenterknowledge.com/archives/2012/10/22/follow-the-moon-scheduling-to-lower-energy-costs/, 2012.

[13] F. Ahmad and T. N. Vijaykumar. Joint optimization of idle and cooling power in data centers while maintaining response time. In *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XV, 2010.

[14] Cullen Bash and George Forman. Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center. In *Proceedings of the USENIX Annual Technical Conference*, ATC'07, 2007.

[15] Eun Kyung Lee, Hariharasudhan Viswanathan, and Dario Pompili. Vmap: Proactive thermal-aware virtual machine allocation in hpc cloud datacenters. In *HiPC*, 2012.

[16] Luiz Andre Barroso and Urs Holzle. The case for energy-proportional computing. *IEEE Computer*, 2007.

[17] Eugen Feller, Louis Rilling, and Christine Morin. Snooze: A scalable and autonomic virtual machine management framework for private clouds. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '12, 2012.

[18] The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 2004.