Parallel Processing Letters

ERIDIS: ENERGY-EFFICIENT RESERVATION INFRASTRUCTURE FOR LARGE-SCALE DISTRIBUTED SYSTEMS

ANNE-CECILE ORGERIE and LAURENT LEFEVRE
École Normale Supérieure de Lyon, INRIA, Université de Lyon
LIP (CNRS, ENS Lyon, INRIA, UCB Lyon)
46 Allée d'Italie, 69364 Lyon Cedex 07, France
annececile.orgerie@ens-lyon.fr - laurent.lefevre@inria.fr

Received January 2011
Revised March 2011
Communicated by Guest Editors

ABSTRACT

At the age of petascale machines, cloud computing and peer-to-peer systems, large-scale distributed systems need an ever-increasing amount of energy. These systems urgently require effective and scalable solutions to manage and limit their electrical consumption. As of now, most efforts are focused on energy-efficient hardware designs. Thus, the challenge is to coordinate all these low-level improvements at the middleware level to improve the energy efficiency of the overall systems. Resource-management solutions can indeed benefit from a broader view to pool the resources and to share them according to the needs of each user. In this paper, we propose ERIDIS, an Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems. It provides a unified and generic framework to manage resources from Grids, Clouds and dedicated networks in an energy-efficient way.

Keywords: Energy efficiency, large-scale distributed systems, green computing, green networking, resource reservation

1. Introduction

Large-scale distributed systems (datacenters, Grids and Clouds) consist in collections of multiple computing and storage resources that communicate through a communication system that can either be shared with other infrastructures or not. The overall energy consumption of such systems is huge and can be split in two parts:

- the fixed part, which depends on the system's size and on the type of each equipment (computing, storing and networking equipments);
- the variable part, which depends on the usage of computing and storage facilities and on the networking traffic.

To reduce the fixed energy costs, equipments can be put into sleep modes when they are not used: for example, computers and cores can be turned off, network parts can be switched off. Likewise, to reduce the variable costs, some slowdown techniques can be used like Dynamic Voltage Frequency Scaling (DVFS) [20] for processors or Adaptive Link Rate (ALR) [6] for Network Interface Cards (NICs). These techniques adjust the processor speed or the transmission rate to the load when the full resource capacity is not required, thus saving energy with only a minor impact on performance. Improvements on the design of hardware component are also desirable to enhance the energy efficiency of each equipment. These techniques can be combined at the system-management level to coordinate the local energy-aware decisions and to tightly couple energy consumption and workload. The ideal goal is to achieve a consumption-workload relation as proportional as possible in a consume-as-you-use manner [5]. Thus, the fixed part of the energy costs will be suppressed, and so will be the wastage due to idle consuming resources.

In-advance reservation mechanisms are widely used in large-scale distributed systems [38, 7, 34] since they guarantee users a certain quality of service, including the respect of deadlines and specific hardware and software constraints, in an infrastructure-as-a-service way. Indeed, users can specify a deadline (when the job should be completed), a start time, and some hardware and software requirements, for example. In-advance reservations also allow a more flexible and predictable resource management: the length of each reservation is known at its submission, thus making the task of scheduling algorithms easier.

We propose an Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems (ERIDIS) in order to optimize the energy used by such systems. The proposed approach deals with fixed and variable parts of the energy consumption. This infrastructure acts at the resource-manager level and includes:

- energy sensors that collect, in real-time, the energy consumption of resources and directly measure the impact of the taken decisions;
- allocating and scheduling algorithms to optimize the reservation placement;
- on/off facilities to put resources into sleep mode when they are not used;
- prediction algorithms to anticipate the workload;
- workload aggregation policies to avoid frequent on/off cycles for the resources.

An application of ERIDIS in three different scenario contexts: data center and Grid resource management, virtualized environments and cloud resource management, and large-data transfers in dedicated networks is proposed and analyzed.

Section 2 deals with the related work. ERIDIS is presented in Section 3. Section 4 details the three applications of this infrastructure: in a Grid and data center context, in a Cloud context, and finally in a network context. Section 5 concludes and presents our perspectives.

2. Related work

Although energy has been a matter of concern for sensor networks and battery-constrained systems since their creation, energy issues are recent for plugged systems. We present related work in the domain of energy-efficiency that can be used in data centers, grids, clouds and wired networks.

2.1. *Server and data center power management*

A huge waste of energy can be observed for various computing and networking equipments: PCs, switches, routers, servers, etc., because they remain fully powered-on during idle periods. In a data-center context, different policies can be applied on different levels: the node level, the cluster level or the network level.

Data centers are made up of a large number of servers with high power requirements concentrated in a small area. They need huge power capacities and the first difficulty is to find out the consumption of all their components (network equipments, nodes, cooling system). In [12], the authors make a model of energy consumption that uses the CPU's activity. A different approach consists in deducing it by using event-monitoring counters [26], for example.

Naturally, the first idea to save energy is to be able to shut down idle nodes [8]. This leads to a problem: how to wake them up when required? Wake-On-LAN is a mechanism implemented on Ethernet cards to allow a distant user to wake up a PC by sending to it some packets via the network [14]. However, such a mechanism requires the Ethernet card to be powered at all times.

Another issue is to have an energy-aware scheduling algorithm to attribute nodes to the tasks, which can be divisible or not [8, 44], requiring synchronization [21], etc. In [9], the authors want to minimize the consumed energy by minimizing the number of joules per operation. The resource manager gets a set of awake nodes and should minimize its size as much as possible. When a task ends on a node, it tries to move the other tasks on this node to the other running nodes. And if a new task arrives, it tries to put it on the awake nodes. The other nodes remain off. This algorithm includes no load balancing mechanisms, so it seems that some nodes will be worn prematurely while others will stay unused.

Other energy-aware resource-management algorithms include load balancing [26] and thermal management concerns [35, 37]. As for all the algorithms presented, the unnecessary wake-ups waste energy twice: by the wake-up power spike and during the idle state on time before going to sleep again. Such algorithms should thus be carefully designed in order not to shut down nodes unnecessarily.

2.2. *Node optimizations*

Energy savings at the node level can also lead to great energy savings with the scale effects. To reduce the wake-up power spike and the booting time, "suspend to disk" techniques can be used. When a node switches to that state, all the content of the

main memory is saved to the hard drive in an hibernate file, preserving the state of the operating system (all the open applications, documents, etc.). All of the node's components are turned off and, at the next state switch, the node will load the hibernate file, restoring the previous state.

Other hardware improvements could be done into the CPU. Some algorithms include DVFS (Dynamic Voltage Frequency Scaling) techniques [21, 20, 12, 44]. The CPU reduces its frequency and voltage when it is underused [20]. These techniques have been already standards on laptops since few years. This increases a lot the range of possible energy savings since they allow to save energy when the nodes are not idle, but not fully used either.

Actually, each computing node is made up of an assembly of several components, and each component can be optimized to save energy. Each component indeed benefits from different states from fully-on to fully-off, which correspond to different energy consumptions. For example, a CPU has one off-state and several on-states which correspond to each possible frequency and scaling on this CPU [20]. Most of the components are the subject of specific studies: NICs [14], disks [2] and CPUs [11] for example. However, the motherboard remains the component consuming the most energy, and it can only be turned off when the entire node can. Thus, IPMI techniques are really useful to remotely and quickly switch nodes on and off [22].

2.3. *Virtualization to improve energy efficiency*

Virtualization is now widely used to provide a large number of computing resources, and the energy consumption of Cloud infrastructures is as problematic as in data centers and grids [27, 40, 41]. Yet, even if virtualization adds a software layer that consumes energy [42], it actually allows finer load consolidation on each node [39], and offers the possibility to use live migration techniques [43] to strengthen load aggregation. Still, these techniques have a cost [33] and should be carefully studied in order to reduce the overall consumption of Clouds.

2.4. *Energy-awareness in wired networking equipments*

The consumption of wired networking devices has been considered for about ten years. In [17], the authors present an interesting approach able to switch off network interfaces, routers and switch components. They first analyze traces and check whether there really are periods of inactivity [17, 16]. Then, they design algorithms to shut down resources based on periodic protocol behavior and traffic estimation [16]. A lot of energy can be saved this way, by running their algorithm onto utilization traces. Then, they propose to save energy even on under-utilization periods. Therefore, they use the low-power modes available on most Ethernet interfaces [19] (that means using Gigabit Ethernet cards at 10 MBps, 100 MBps or 1 GBps). Their results show that their algorithm does not affect the communication performance in terms of both delay and packet loss.

Their algorithms are based on predictions to take sleeping decisions. They use buffer occupancy, the behavior of previous packet (arrival times) and a maximum bounded delay [18]. They assume that the routers are able to store packets in their buffer even if they are asleep. When the buffer occupancy reaches a certain size, they wake up the whole router.

The real problem of shutting down networking devices is how to ensure network presence. Indeed, when a switch is asleep, it may not answer to the requests (ARP requests or PING for example). Moreover, normally when a link is re-established, an auto-negotiation protocol is run (to synchronize clocks, determine link type and link rate, etc.) and this takes about few hundreds of milliseconds, which is too long on high-capacity links. To address this, they modify the auto-negotiation protocol for their algorithms [18]: the auto-negotiation is not run after sleeping periods because those are really short and the link has not changed state during them.

Another solution is to use proxying techniques: the Ethernet card or the switch filters packets that require no response (like broadcasts), replies to packets that require minimal response (like ping) and only wakes up the system for packets requiring a non-trivial response [14]. In [14], the authors give a complete analysis of the traffic received by an idle PC and they explain that most of this traffic would be filtered-out or trivially responded to by a proxy.

These authors have also proposed an algorithm called Adaptive Link Rate (ALR) which changes the link's data rate based on an output buffer threshold policy [15]. This algorithm does not affect the mean packet delay.

2.5. *Synthesis*

Lots of computing and networking equipments are concerned by these overall observations on the waste of energy: PCs, servers, storage devices, switches, routers, etc. In a context of large-scale distributed systems, different policies can be applied depending on where users want to make savings: at the node level, the data-center level, or the network level.

At the node level, one can use Direct Voltage Scaling techniques and frequency-scaling techniques to reduce the energy consumption of the CPU. But one can also imagine putting into sleep cores, memory banks or disks for example. Indeed, computers are increasingly multi-core and multi-bank. So it should be possible to turn on or shut down one core or one bank upon request.

At the data center level, different solutions are also possible to reduce energy consumption such as energy-efficient task scheduling, proxying techniques to ensure network presence or resource virtualization. With the scale effect, the potential savings are huge.

At the network level, we can consider the possibility to shut down entire redundant routes with all their components. But we can also shut down, one by one, the interfaces of the routers, switches and computers or just scale their speed.

3. ERIDIS: Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems

Different techniques are available to improve the energy efficiency in large-scale distributed systems, but they require a unified framework to associate them. In order to reach this goal, we propose an Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems called ERIDIS.

3.1. *ERIDIS architecture*

With the growing number of deadline-driven applications such as meteorological disaster or earthquake forecast, or important large data-transfers (e.g., for daily news), reservation infrastructures for large-scale distributed systems are increasingly important. Our Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems (ERIDIS) strives to optimize energy consumption in such systems.

The premise behind ERIDIS is that parts of the computing and networking resources can be put in a sleep-state when not needed to execute user tasks in order to consume less energy. Through coordinated resource management, both the opportunity for allowing resources to go to sleep and the duration of these sleeping events can be increased.

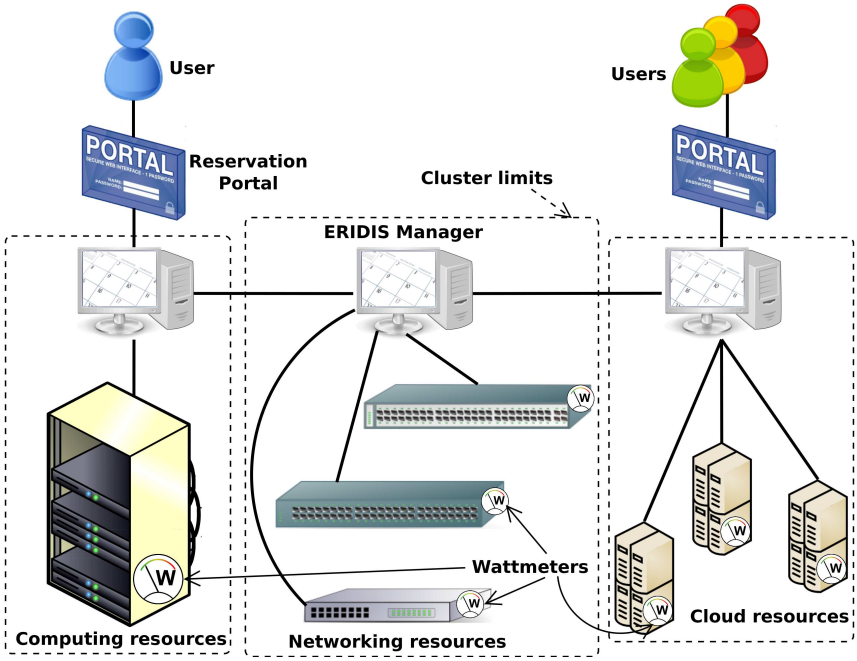

Fig. 1. ERIDIS components

Depending on whether they have a computation or a data transfer to be done,

users make reservation requests that consist of, at least:

- an earliest possible start time, a deadline, a number of computing resources required, and a duration in the case of a computing job, or
- an earliest possible start time, a deadline, a data volume to be transferred (in GB for example), a source node, and a destination node in the case of a networking job.

Other precisions can be included as a particular computing environment or a particular geographic location to use for a computing job, or a bit-rate transmission profile (maximum possible bit-rate over time) for a networking job. These particular additional options will be detailed in the next sections.

Figure 1 presents the logical architecture of ERIDIS: users are connected to a reservation portal, which is the ERIDIS gateway for them. Each portal is directly connected to an ERIDIS manager, which is a local resource manager and is in charge of the management of a whole cluster of resources (either computing, Cloud or networking resources). For example, for a Grid infrastructure, each cluster (e.g., Grid site) has its own *ERIDIS resource manager* to keep the overall architecture scalable, fast and robust. So, users send their requests to their local gateway, which is part of the resource manager, and this latter is in charge of dealing with the other resource managers, if required.

Each resource managed by ERIDIS is monitored by energy sensors (wattmeters) which provides accurate and periodical measurements to the manager. Thus, at the end of each reservation, ERIDIS is able to compute the energy consumption of this particular reservation, and can provide this information to the user in order to increase energy-awareness.

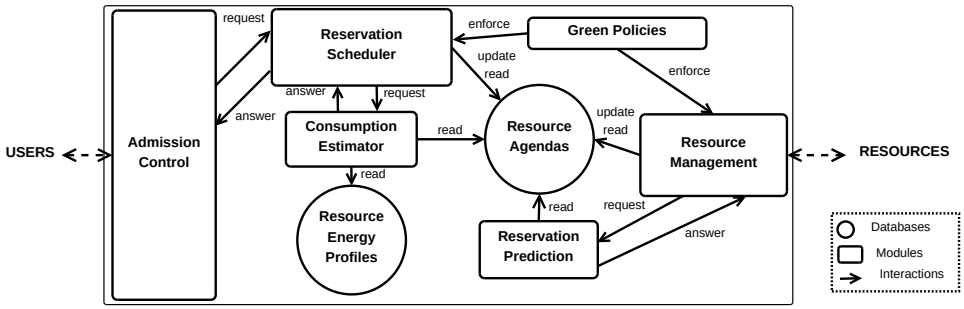

Fig. 2. Structure of an ERIDIS manager

The structure of an ERIDIS manager is presented in Figure 2. Administrators and system designers can specify the green policies they want to use: on/off techniques, DVFS techniques, etc. These choices are taken into account in the reservation scheduler and in the resource-management modules. The reservation scheduler is in charge of scheduling the incoming reservation requests and allocating resources

to them. The resource management module is responsible for putting resources into sleep state or waking them up if required. All the components of this manager are going to be described in details in parallel with the explanation of the working of ERIDIS.

3.2. *The reservation model*

Each ERIDIS manager maintains an *agenda* for each resource that it is in charge of (Resource Agendas on Figure 2). An agenda stores all the future reservations concerning the resource. So, it may contain different *resource states*: powered on but idle, turned off, booting, switching off, or partially or fully reserved, with the duration spent in each state. The resource-management module is responsible for updating the agendas if some resources are down.

An example of such a resource agenda is shown on Figure 3. This agenda contains two reservations and the resource is switched off between these two reservations and again after the second reservation. The date of a change in resource state is called an *event*. The capacity can be the number of core in the case of computing resources or the bandwidth in the case of networking resources.

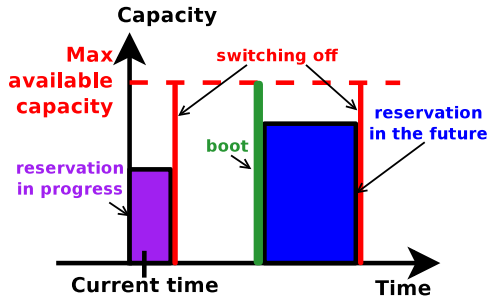

Fig. 3. Example of an agenda of a given resource

This model uses a continuous time model in order to have the storage of the agendas be more flexible and less space-using. The storage of the agendas is structured in *time-capacity list* which is made of $(t[i], c[i])$ tuples, where $t[i]$ represents an event (time) and $c[i]$ is the percentage of the resource that is used from that event to the next one. For example, for a bandwidth reservation, the capacity is the reserved bandwidth portion, and for a computing job, the capacity represents the percentage of CPU which has been reserved for this job. Thus, at each time, a resource knows which percentage of its overall capacity is used. Some particular values are used for the capacity to represent the power-on and shut-down periods.

These tuples are sorted in increasing order of $t[i]$. If $(t[n], c[n])$ is the last tuple, then it means that the resource is unused from $t[n]$ to ∞ ($c[n] = 0$). The ERIDIS manager is in charge of updating the agendas of the resources it is managing. Yet,

each resource stores its own agenda to know when it should go to sleep, and when it should wake up.

As a matter of fact, the architecture of ERIDIS is semi-decentralized since each local manager gets full control of the resources it manages. Moreover, local managers discuss among themselves in a point-to-point manner without a global supervisor on top of them. For example, in a Cloud context, if a user requires more resources than its local manager can offer him, then the local manager will contact another manager to request the number of resources that it lacks. If the number of available resources is still insufficient, the local manager will ask another manager in addition and so on, until it depletes the whole list of resource managers. This is why each ERIDIS manager is not necessarily directly linked to a user portal. When a new manager is added, it informs all the existing managers, and they update their manager list.

This semi-decentralized feature ensures that the architecture is scalable, fast and robust since any ERIDIS manager is in charge of a limited number of resources and has a privileged access to the other ERIDIS managers.

3.3. *Management of the resource requests*

A reservation is like a lease contract between the resource provider and the resource user: the user can use the reserved resources during a limited time interval which has been fixed during the negotiation process between the user and the provider. The negotiation takes place in a three-steps handshake process (Figure 4).

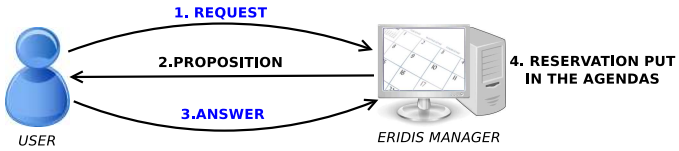

Fig. 4. Reservation negotiation between the user and an ERIDIS manager

First, the user sends a *request* which contains its resource and time constraints, i.e., at least the type of resources, the required number, the reservation duration, and a deadline. Secondly, the reservation manager proposes a solution, which can either fulfill the user's requirements if it is possible or propose the earliest possible reservation start time if the deadline constraint cannot be respected (due to system load). If the request is not acceptable because it asks for too many resources, or because it does not respect the system's admission rules, the request is rejected and it is notified to the user.

Finally, if the reservation manager has made a proposition, the user has a fixed amount of time to answer it and express its agreement or disagreement. Otherwise, the request is rejected. If the user accepts the manager's proposition, the manager updates all the concerned agendas by adding this new reservation. Algorithm 1 details the algorithm used to process the request. The scheduling algorithm will be

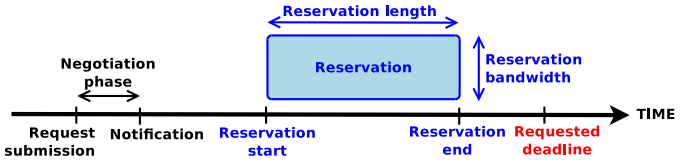

Fig. 5. Reservation process

detailed later (Algorithm 2).

The reservation-management system should guarantee that the negotiated reservation terms are well respected. It also has to dynamically schedule in time the reservations and allocate them the most appropriate resources. The sequence of operations is described on Figure 5.

Algorithm 1 Request-Processing

```
if the request is not acceptable then
  reject it
else
  launch the scheduling algorithm
  if the request cannot be satisfied then
    propose the earliest possible start time (given by the scheduling)
  else
    send the offer to the user (given by the scheduling)
  end if
  wait for the answer
  if there is no answer or the answer is no then
    discard the request
  else
    place the reservation on each concerned agenda
  end if
end if
```

Figure 6 presents the different reservation states. First, the user sends a request which is either rejected or acceptable, and in that case, the manager makes a proposition. Then, this proposition is either accepted or rejected by the user. If it is accepted, the reservation is scheduled. This reservation can be re-scheduled before its start time if the manager needs to move this reservation to accept a new one. This re-scheduling is still bound by the resource and time constraints negotiated between the user and the ERIDIS manager. No re-negotiation is allowed in our model. During the reservation, the user has access to the reserved resources until the reservation's end time.

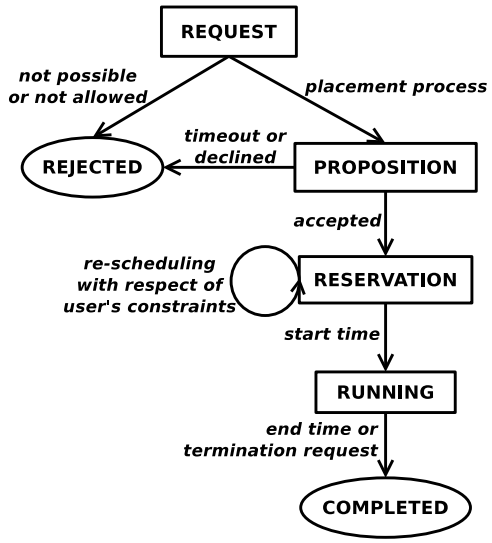

Fig. 6. Management of a reservation: the different states

3.4. *Energy-efficient Reservation Scheduling*

The scheduling algorithm should provide either the most energy-efficient possibility in terms of both time placement and resource allocation, or the earliest possible start time if the deadline constraint cannot be satisfied. This algorithm is executed after the admission control process, that is to say that it is executed on requests that respect the system's rules and that can be accepted depending on the system's load.

If the reservation is possible, the scheduling algorithm provides a reservation start time and a set of resources to be used during the reservation. Algorithm 2 details the scheduling method. The first step is to find the earliest possible start time. As the request has passed the admission control, it is acceptable and thus, the user constraints in terms of resources can be satisfied at a time or another. So, this first step will determine if the deadline constraint can be satisfied or not. Indeed, if the earliest possible start time does not allow the reservation to end before the deadline because of the system load, this start time is directly proposed to the user which can either accept or decline this proposition.

The earliest possible start time is found by looking linearly at each event from the current time to see if a reservation satisfying the user's constraints can be placed before or after this event. Only the feasible solutions are examined, the algorithm does not put a reservation start time before the current time for example. This restriction of the solution space to certain dates (i.e., the events) ensures that the algorithm is fast and scalable and that the found solution is energy-efficient, since the reservation has been aggregated with at least another one (by definition of the events).

Algorithm 2 Scheduling-Algorithm

```
find d the earliest possible start time with a satisfying set r of free resources
List = [(d, r)]
for all agenda events between d and (deadline - reservation duration) do
  if the reservation can be placed before this event then
    find the set of the least consuming free resources at that time
    add the possible start time to List with the resource set
  end if
  if the reservation can be placed after this event then
    find the set of the least consuming free resources at that time
    add this event to List with the set of resources
  end if
end for
if List contains only d then
  return d and r
else
  for all dates in List do
    estimate the reservation energy consumption if it starts at this date with the
    corresponding set of resources
  end for
  propose the less energy consuming date and set of resources to the user
end if
```

When the earliest possible d has been found, the scheduling algorithm looks linearly at all the resources which can accept a reservation starting at d, and it picks the N least consuming resources at that time (taking into account the necessary switching on and off), where N is the number of resources required by the user. The energy-consumption estimation of a reservation is detailed in Section 3.6 (Consumption Estimator module on Figure 2). This set of N resources should satisfy the user's constraints (e.g., if he or she specified some particular hardware or software constraints), and should be the least consuming set of resources among all the possible sets at d.

When the earliest possible solution (d and r the set of resources) has been found, the algorithm (Algorithm 2) searches if there is a better solution, in terms of energy consumption, after d but still before the deadline. So, the algorithm examines each event in the time interval between d and the deadline minus the reservation duration requested by the user. The process is similar to the one used to find d and r. However, each event of the time interval is considered, and for each event, the set of the N least consuming resources is determined. In the end, all these solutions are compared in term of energy consumption and the best one is chosen. This solution is optimal, since the algorithm tests all the possibilities that can be the less consuming ones due to the energy-saving properties of reservation aggregation. The resources

are shared with other reservations, thus the optimal set of resources depends on the considered start time. As the available capacity of each resource varies over time (due to reservations), for each possible start time, the energy consumption estimation is used to pick the best solution.

3.5. *Resource management and on/off algorithms*

The scheduling algorithm aims at aggregating as much as possible the reservations in order to save energy, and especially the energy used to switch the resources on and off. However, switching the resources on and off can be difficult and requires time and energy. Thus, when a resource is switched off, it should stay off for a certain amount of time, to save more energy than is being used when switching it off then on again. Switched-off resources consume energy, but less energy than when they are idle (powered on but not in use) [33]. Thus, a subtle balance needs to be adopted to ensure energy savings by switching resources off. That is why, at the end of each reservation, the resource manager determines if the freed resources should remain on or should be switched off.

Algorithm 3 Reservation-End

```
Provide the energy consumed by the overall reservation to the user
for all reserved resources do
  if this resource has an imminent reservation then
    let it on
  else
    launch the prediction algorithm
    if the resource is going to be used soon then
      let it on
    else
      turn it off
    end if
  end if
end for
```

First, for each resource, the manager examines the resource agenda to see whether another reservation has been put just after it. If it is the case, the resource stays on. Otherwise, the manager uses a prediction algorithm (detailed in Section 3.7) to estimate whether the resource is going to be used soon (Reservation Prediction module on Figure 2). If it is the case, it stays on; otherwise, the resource is switched off. This process is described by Algorithm 3. An *imminent* reservation is a reservation that will occur in short period of time such as it is more costly to shut down the resource and to turn it back on than let it idle.

The problem here is to have a reliable and fast solution to remotely switch the resources on and off when necessary. This requires some hardware facilities on the

resources and a dedicated infrastructure to access them. For example, in a Grid or Cloud context, IPMI facilities can be used [22].

Moreover, each resource has a copy of its own agenda and thus, when it goes into sleep state, it uses a timer to know when it should wake up again. So, the previous mechanism to remotely switch on and off the resources is used only when a reservation, which was not planed when the resource went to sleep, occurs. Yet, switching operations on and off should be fast enough to avoid impacting the reactivity of the whole system.

3.6. *Energy-consumption estimates*

As seen on the architecture, ERIDIS embeds energy sensors to monitor in real-time the energy consumption of the resources it manages. So, at the end of each reservation, the ERIDIS manager reports the energy consumption of the overall reservation to the user, to increase his energy awareness (Algorithm 3).

As we have seen previously, estimations of the energy consumption of some reservations are needed to take the best scheduling decisions. To make these estimations, the ERIDIS manager requires an energy profile for each resource it is managing. This energy profile contains information about:

- the energy and time required to switch the resource off,
- the energy and time required to switch the resource on,
- the mean power used by the resource when it is off (i.e., sleep state),
- the mean power used by the resource when it is idle (i.e., powered on but not used),
- the relation between the capacity usage (in percentage) and the mean power usage of the resource (function giving the power consumption as a function of the percentage utilization of the resource).

The ERIDIS manager is responsible for drawing up these energy profiles by using benchmarks and the energy sensors (Resource Energy Profiles on Figure 2). These profiles are periodically updated (each month, for example) because hardware usage ("wear and tear"), heat or humidity conditions, among others, can impact the energy consumption of the resources.

By exploiting these energy profiles, it becomes easy for the resource manager to estimate the energy consumption of a reservation. This estimation includes the energy cost to wake up and to switch off resources if it is necessary. Thus, if for example the reservation is aggregated after another one, the energy cost to wake up the resources is saved. Similarly, if a reservation shares some resources with another one because it does not require the full resource, the working energy cost of the resources is split between the two reservations according to the percentage of the resource that each of them is using.

3.7. *Prediction algorithms*

As explained earlier, prediction algorithms are used to ensure a good planning of the off-on cycles. Our prediction algorithms are based on the recent history (the past part of the agenda). They are based on average values of past inactivity period durations and feedbacks which are average values of differences between the past predictions and the past corresponding real events in the agenda.

For example, with computing resources, when a node is freed, the average value of the last few free intervals of time (when the node is not used) is computed, and it is assumed to be the value of the next free interval of time. This prediction algorithm has been tested in [31] with usage traces of a cluster belonging to French experimental platform Grid'5000. In about 70% of the cases, the algorithm takes the good decision between switching off the resource and leaving it on. More generally, the energy gain is significant.

3.8. *Energy consumption optimization: re-planning capacity*

We have presented algorithms which work in a static way: once a decision has been taken, it cannot be changed. Our re-planning functionality is dynamic and allows reservations to be moved after their registration in the agendas while still guaranteeing the same QoS and respecting the user deadline. Indeed, off-line algorithms can lead to optimal solutions in terms of energy conservation because all the reservation requests are known from the beginning. It is not the case with on-line algorithms such as ours. However, off-line algorithms are not appropriate for real-time scheduling, so this re-planning capacity is a good trade-off between on-line and off-line scheduling.

An example of how this re-planning functionality works is presented on Figure 7 with the agendas of two different resources before and after the re-planning operation. Reservation $R4$ has just been stored into the two agendas and thus reservation $R3$ can be put just after it if it is more energy efficient than the former solution. This re-scheduling algorithm is launched periodically.

To illustrate how ERIDIS works, three use cases will be presented: grid management (Section 4.1), cloud management (Section 4.2), and data transfer in a network overlay (Section 4.3).

4. Use cases

ERIDIS is a generic infrastructure we applied it to three different contexts:

(1) data-center and grid resource management with the Energy-Aware Reservation Infrastructure (EARI),
(2) virtualized environments and cloud resource management with the Green Open Cloud (GOC),
(3) data transfers in dedicated networks with the High-level Energy-awaRe Model for bandwidth reservation in End-to-end networkS (HERMES).

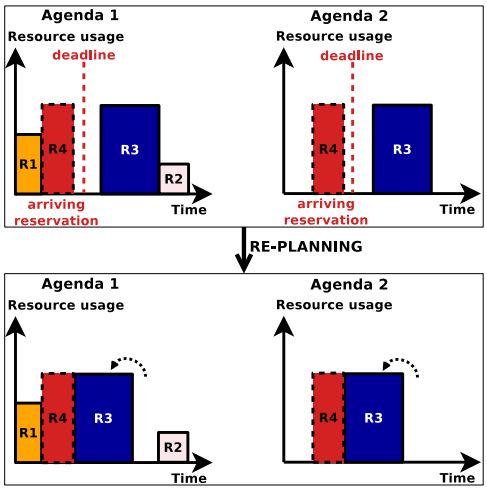

Fig. 7. Re-planning capacity

4.1. *Energy-Aware Reservation Infrastructure (EARI)*

The energy consumption of data centers worldwide has doubled between 2000 and 2006 [25]. The incremental US demand for data center energy between 2008 and 2010 is the equivalent of ten new power plants [25]. These alarming figures lead to think about new technologies and infrastructures in order to increase the energy efficiency of large-scale distributed systems such as Data Centers, Grids and Clouds.

The main leverage to make large-scale distributed infrastructures more energy-efficient is to reduce energy wastage. Indeed, resources are always fully powered on even when they are not used. So, Grids require energy-aware frameworks capable of switching unused resources off without impacting user applications in terms of both performance and usage. This is why, we propose the Energy-Aware Reservation Infrastructure (EARI) [31, 32, 23] based on the ERIDIS model.

EARI is devoted to Grid infrastructures that support in-advance reservations: users submit reservation requests. They specify the duration, the number of resources and the start time they want. When a reservation is accepted, the scheduler puts it in its agenda and cannot move it afterwards.

Figure 8 presents the architecture of EARI for a cluster. It is composed of a traditional Data Center infrastructure: users, a portal, a scheduler and resource manager, and the Grid resources. However, it is also composed of energy-aware components: a set of energy sensors plugged to the resources and an energy-aware manager that is responsible for applying the green policies of EARI.

All the resources managers are linked and communicate between them in order to satisfy user requests. The global EARI is evaluated in [32] by simulating a replay

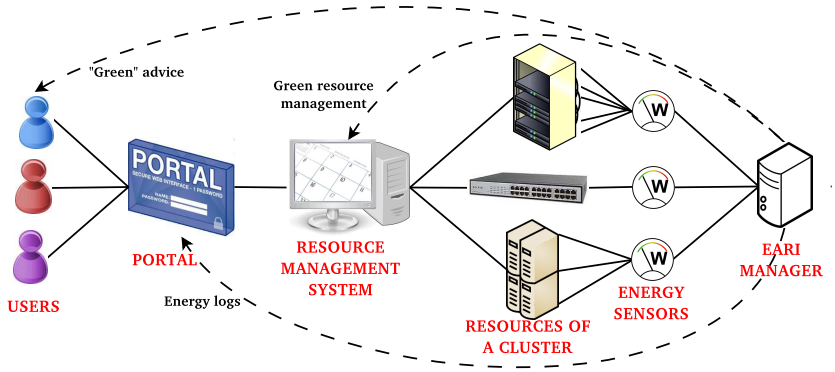

Fig. 8. EARI components

of one year of logs of Grid'5000[a]. On the Lyon site of Grid'5000 (150 nodes), we have deployed energy sensors that fully monitor the site [10]. This provides an experimental testbed where we can test our frameworks with real energy measurements.

To evaluate EARI, we conducted experiments based on a replay of the 2007 traces of the Grid'5000 platform (these traces have been studied in [31]). We found that using EARI can lead to energy gains up to 46% depending on the cluster's load [32].

4.2. *Green Open Cloud (GOC)*

Among the Cloud's most well-known features are: virtualization, accounting, scalability, reliability and security. The Resources-as-a-Service (RaaS) philosophy leads to a more flexible management of the physical nodes: Clouds provide a strong isolation that allows users to share the same physical resources. Thus, this strong virtual machine (VM) isolation can also lead to energy savings. Indeed, physical resources can be exploited more using workload consolidation.

For this reason, we adapted ERIDIS to Cloud environments in order to benefit from the Cloud's features. Still, some differences between Grids and Clouds have to be taken into account:

- agenda (resources can be shared in a more flexible manner),
- virtualization and possibility to use live migration,
- usage and thus predictions,
- resource management.

Cloud computing seems to be a promising solution to the increasing demand

[a]Grid'5000 is a French experimental platform comprising 5000 cores distributed in 9 sites in France and linked with a high-speed dedicated network. Grid'5000 is an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (http://www.grid5000.fr).

of computing power needed by more and more complex applications. However, the studies often lack real values for the electric consumption of virtualized infrastructures. That is why, as a preliminary step, we have studied the energy consumption of virtual machines completing basic operations: boot, shut down, doing a cpu-burn task, and migrating [28, 24].

These analyzes led us to propose the Green Open Cloud (GOC) framework to manage Cloud resources in an energy-efficient way. Among the components of a Cloud architecture, we have decided to focus on virtualization, which appears as the main technology used in these architectures. We also use migration to dynamically unbalance the load between the Cloud nodes in order to shut down some nodes, and thus to save energy.

As EARI, GOC supports the "do the same for less" approach, and deals with energy-efficient on/off models combined with prediction solutions [28, 24]. We have evaluated GOC on nodes with real measurements provided by external wattmeters. We found that, on the studied scenario, we can save up to 25% energy by using GOC [24].

4.3. *High-level Energy-awaRe Model for bandwidth reservation in End-to-end networkS (HERMES)*

Scientists increasingly rely on the network for high-speed data transfers, results dissemination and collaborations. Networks are thus becoming the critical component. In 2007, to distribute the entire collection of data from the Hubble telescope (about 120 terabytes) to various research institutions, scientists chose to copy this data on hard disks and to send these hard disks via mail. It was faster than using the network [13]. To solve this issue, dedicated networks were built to transfer large amounts of scientific data, e.g., for the LHC (Large Hadron Collider) which produces 15 million gigabytes of data every year [1].

Bandwidth provisioning is feasible for network operators for several years thanks to protocols such as MultiProtocol Label Switching (MPLS) [36] and Reservation Protocol (RSVP) [45]. However, for end users with no knowledge of network traffic, this task is impossible without collaboration with the other nodes.

On the other hand, as networks become increasingly essential, their electric consumption reaches unprecedented peaks [4]. Up to now, the main concern to design network equipments and protocols was performance only; energy consumption was not taken into account. With the costly growth in network electricity demand, it is high time to consider energy as a main priority for network design.

To this end, we propose a new complete and energy-efficient bulk data transfer (BDT) framework including scheduling algorithms which provide an adaptive and predictive management of the advance bandwidth reservations (ABR). This model is called HERMES: High-level Energy-awaRe Model for bandwidth reservation in End-to-end networkS. It is adapted from ERIDIS to this particular dedicated network case with bulk data transfers.

To achieve energy-efficiency, HERMES combines several techniques:

- unused network components are put into sleep mode;
- energy optimization of the reservation scheduling through reservation aggregation;
- minimization of the control messages required by the infrastructure;
- usage of DTN to manage the infrastructure;
- network-usage prediction to avoid too frequent on/off cycles.

Wired networks are usually not organized into clusters of resources as Grids and Clouds naturally are. However, this kind of self-organization is used since a long time in ad-hoc networks for routing and energy-management purposes [3]. Network nodes are divided into virtual clusters with a cluster head managing each cluster. In our case, the cluster head is the HERMES manager. This manager is responsible for the agendas of the nodes belonging to its cluster.

The architecture of HERMES is shown on Figure 9. Here, the user portal to access the reservation system is called a *gateway*. The network case is particular because it requires the coordinated reservation of several resources at the same time, but for a particular request, this number of resources is not known in advance. Indeed, for a data-transfer job, several network paths may be possible, and they may not have the same length. That is why, in a first approach, we have chosen to fix the network paths between each source-destination pair. So, we first use a Dijkstra-based algorithm to find two shortest paths for each pair (with the less number of links in common). Then, the scheduling algorithm is launched separately on these two paths and the least energy consuming solution is picked.

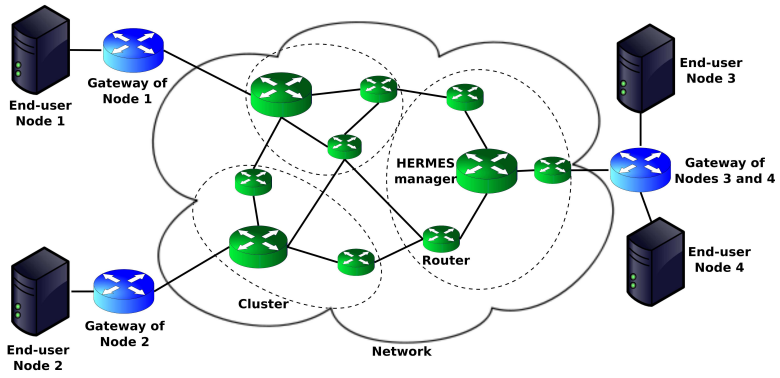

Fig. 9. Organization in clusters with HERMES

A detailed architecture of HERMES and discussion about the validity of this model is provided in [29]. Simulation results show that two-thirds of the energy used by current data center, grid and cloud networks could be saved using HERMES [30].

5. Conclusion

Due to their size and heterogeneity, large-scale distributed systems require scalable, robust, fault-tolerant and energy-efficient resource-management infrastructures. This paper presents ERIDIS: an Energy-efficient Reservation Infrastructure for large-scale DIstributed Systems. ERIDIS is empowered to optimize the energy consumption of the computing and networking resources and to have a flexible and adaptive reservation management which satisfies user requirements through strict reservation policies. It provides a unified framework to manage these heterogeneous resources in an energy-efficient way.

As a proof of concept, we have adapted ERIDIS to three different application fields to demonstrate its capabilities:

- data-center and grid management with the Energy-Aware Reservation Infrastructure (EARI),
- virtualized environments and cloud management with the Green Open Cloud (GOC),
- data transfers in dedicated networks with the High-level Energy-awaRe Model for bandwidth reservation in End-to-end networkS (HERMES).

Our next step is to refine the ERIDIS model to take advantage of divisible tasks that can be suspended and resumed as required if some more urgent tasks need to be executed. These tasks allow a higher level of flexibility in the reservation management.

References

[1] http://lcg.web.cern.ch/lcg/public/default.htm.
[2] M. Allalouf, Y. Arbitman, M. Factor, R. Kat, K. Meth, and D. Naor. Storage modeling for power estimation. In *SYSTOR 2009: The Israeli Experimental Systems Conference*, pages 1–10, 2009.
[3] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Network*, 7(3):537–568, 2009.
[4] M. Baldi and Y. Ofek. Time For A "Greener" Internet. In *GreenCom: International Workshop On Green Communications (in conjunction with the IEEE ICC)*, 2009.
[5] L.A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33 –37, 2007.
[6] M. Bennett, K. Christensen, and B. Nordman. Improving The Energy Efficiency Of Ethernet: Adaptive Link Rate Proposal. Ethernet Alliance White Paper, 2006.
[7] C. Castillo, G.N. Rouskas, and K. Harfoush. Efficient resource management using advance reservations for heterogeneous Grids. In *International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–12, 2008.
[8] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing energy and server resources in hosting centers. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 103–116, 2001.
[9] J. Chase and R. Doyle. Balance of Power: Energy Management for Server Clusters. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS'01)*, 2001.

[10] M. Dias de Assuncao, J.-P. Gelas, L. Lefèvre, and A.-C. Orgerie. The Green Grid'5000: Instrumenting a Grid with Energy Sensors. In *International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID)*, 2010.

[11] H.G. Dietz and W.R. Dieter. Compiler and runtime support for predictive control of power and cooling. In *Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[12] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 13–23, 2007.

[13] C. Farivar. Google's Next-Gen of Sneakernet. [online] http://www.wired.com/science/discoveries/news/2007/03/73007, 2007.

[14] C. Gunaratne, K. Christensen, and B. Nordman. Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed. *International Journal of Network Management*, 15(5):297–310, 2005.

[15] C. Gunaratne and S. Suen. Ethernet Adaptive Link Rate (ALR): Analysis of a Buffer Threshold Policy. In *Global Telecommunications Conference, 2006. GLOBECOM '06*, 2006.

[16] M. Gupta, S. Grover, and S. Singh. A Feasibility Study for Power Management in LAN Switches. In *ICNP '04: Proceedings of the Network Protocols, 12th IEEE International Conference*, pages 361–371, 2004.

[17] M. Gupta and S. Singh. Greening of the Internet. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26, 2003.

[18] M. Gupta and S. Singh. Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links. *IEEE International Conference on Communications (ICC'07)*, pages 6156–6161, 2007.

[19] M. Gupta and S. Singh. Using Low-Power Modes for Energy Conservation in Ethernet LANs. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2451–2455, 2007.

[20] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a PC cluster. In *Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[21] R. Jejurikar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real-time systems. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1024–1037, 2006.

[22] C. Leangsuksun et al. IPMI-based Efficient Notification Framework for Large Scale Cluster Computing. In *International Symposium on Cluster Computing and the Grid Workshops (CCGrid)*, volume 2, 2006.

[23] L. Lefèvre and A.-C. Orgerie. Towards Energy Aware Reservation Infrastructure for Large-Scale Experimental Distributed Systems. *Parallel Processing Letters*, 19(3):419–433, 2009.

[24] L. Lefèvre and A.-C. Orgerie. Designing and Evaluating an Energy Efficient Cloud. *Journal of SuperComputing*, 51(3):352–373, 2010.

[25] McKinsey & Company. Revolutionizing Data Center Efficiency. Technical report, 2009.

[26] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. *SIGOPS Operating Systems Review*, 40(4):403–414, 2006.

[27] R. Nathuji and K. Schwan. VirtualPower: coordinated power management in virtualized enterprise systems. In *Symposium on Operating systems principles (SOPS)*, pages 265–278, 2007.

[28] A.-C. Orgerie and L. Lefèvre. When Clouds become Green: the Green Open Cloud

Architecture. In *Parco: International Conference on Parallel Computing*, 2009.

[29] A.-C. Orgerie and L. Lefèvre. Energy-Efficient Bandwidth Reservation for Bulk Data Transfers in Dedicated Wired Networks. *Journal of Supercomputing, Special Issue on Green Networks*, 2011.

[30] A.-C. Orgerie and L. Lefèvre. Energy-Efficient Framework for Networks of Large-Scale Distributed Systems. In *IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2011.

[31] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Chasing Gaps between Bursts : Towards Energy Efficient Large Scale Experimental Grids. In *PDCAT : International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 381–389, 2008.

[32] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Save Watts in your Grid: Green Strategies for Energy-Aware Framework in Large-Scale Distributed Systems. In *ICPADS: IEEE International Conference on Parallel and Distributed Systems*, pages 171–178, 2008.

[33] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Demystifying Energy Consumption in Grids and Clouds. In *Work in Progress in Green Computing, IGCC Workshop*, 2010.

[34] C. Palansuriya, M. Buchli, K. Kavoussanakis, A. Patil, C. Tziouvaras, A. Trew, A. Simpson, and R. Baxter. End-to-End Bandwidth Allocation and Reservation for Grid applications. In *Conference on Broadband Communications, Networks and Systems (BROADNETS)*, pages 1–9, 2006.

[35] C. Patel, R. Sharma, C. Bash, and S. Graupner. Energy Aware Grid: Global Workload Placement based on Energy Efficiency. Technical report, HP Laboratories, 2002.

[36] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, http://www.ietf.org/rfc/rfc3031.txt, 2001.

[37] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase. Balance of Power: Dynamic Thermal Management for Internet Data Centers. *IEEE Internet Computing*, 9(1):42–49, 2005.

[38] B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster. Resource Leasing and the Art of Suspending Virtual Machines. In *Conference on High Performance Computing and Communications (HPCC)*, pages 59–68, 2009.

[39] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Conference on Power aware computing and systems (HotPower)*. USENIX Association, 2008.

[40] J. Stoess, C. Lang, and F. Bellosa. Energy management for hypervisor-based virtual machines. In *USENIX Annual Technical Conference (ATC)*, pages 1–14, 2007.

[41] R. Talaber, T. Brey, and L. Lamers. Using Virtualization to Improve Data Center Efficiency. Technical report, The Green Grid, 2009.

[42] J. Torres, D. Carrera, K. Hogan, R. Gavalda, V. Beltran, and N. Poggi. Reducing wasted resources to help achieve green data centers. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–8, 2008.

[43] F. Travostino et al. Seamless live migration of virtual machines over the MAN/WAN. *Future Generation Computer Systems*, 22(8):901–907, 2006.

[44] L. Wang, G. von Laszewski, J. Dayal, and F. Wang. Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS. In *Conference on Cluster, Cloud and Grid Computing (CCGrid)*, pages 368–377, 2010.

[45] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7:8–18, 1993.