

*State of the Art of Power Saving in Clusters and
Results from the EDF Case Study*

Eugen Feller — Christine Morin — Daniel Leprince

N° 7473

December 2010

Thème NUM



*Rapport
de recherche*

State of the Art of Power Saving in Clusters and Results from the EDF Case Study

Eugen Feller*, Christine Morin* , Daniel Leprince†

Thème NUM — Systèmes numériques
Équipes-Projets MYRIADS

Rapport de recherche n° 7473 — December 2010 — 39 pages

Abstract: Energy management for mobile devices has been traditionally a well studied topic during the last two decades, as these devices usually do not have a permanent connection to the power grid and thus solely rely on the limited battery charge. However, this trend has been mostly disregarded in the context of HPC systems as the main focus mainly relied on improving the performance at any cost. Therefore, energy costs for operating and cooling the equipment of current data centers have increased significantly up to a point where they are able to surpass the hardware acquisition costs.

In this work we survey the current energy conservation efforts in distributed systems. Hence, we start our work with the introduction of the basic principles of power and energy management. This involves the distinguishing between the two often misleading terms *power* and *energy* management. Furthermore, we present several approaches on how to measure the power consumption at system and component level. Moreover, in order to get a deeper understanding on where the most of the power is spent within a server, we take a closer look at its components (i.e. CPU, RAM, etc.) and outline the currently available energy-saving mechanisms. Given that energy consumption strongly depends on the workload characteristics we also discuss the different types of workloads (i.e. mobile, commercial and scientific). Afterwards, we continue with the current approaches for energy savings in distributed systems by focusing on both *node* and *cluster*-level efforts. Finally, we finish this survey with an outline of some open challenges and the introduction of the *EcoGrappe* project. Thereby, we detail its objectives and present some of the work done at our project partner EDF.

Key-words: Energy saving, Resource management, Green computing, Cloud computing, Virtualization, Dynamic adaptation

* INRIA Centre Rennes - Bretagne Atlantique, Campus universitaire de Beaulieu, 35042 Rennes, France - {Eugen Feller, Christine Morin}@inria.fr

† Électricité de France (EDF) R&D, 1 avenue du Général de Gaulle, 92140 Clamart, France - Daniel.Leprince@edf.fr

État de l'art de l'économie d'énergie dans les grappes de calcul et résultats de l'étude de cas EDF

Résumé : La gestion de l'énergie pour les périphériques mobiles a beaucoup été étudié au cours des deux dernières décennies : les mobiles n'ont comme puissance énergétique que leur seule batterie et n'ont généralement pas de connexion permanente vers les puissantes grilles de calcul. Cependant, ce domaine n'a jamais été étudié de manière approfondie dans le contexte du calcul à haute performance (HPC), le principal objectif étant fondé sur le critère de performance. Ainsi, le coût en énergie dépensé pour le fonctionnement et le refroidissement des équipements informatiques des centres de calcul a augmenté de manière significative, jusqu'au point où celui-ci en devient plus élevé que le coût d'acquisition des systèmes informatiques eux-mêmes.

Ce travail présente un état de l'art traitant de la maîtrise des dépenses énergétiques dans les systèmes distribués. Nous commençons notre étude en introduisant les fondamentaux et les différences liés à la gestion de la *puissance* et de l'*énergie*, deux termes très souvent confondus. De plus, nous proposons différentes approches pour mesurer la consommation d'énergie des systèmes complet (par exemple, un serveur) et des composants physiques le constituant. En outre, afin de comprendre précisément où est dépensé la majorité de l'énergie consommée par un serveur, nous étudions ses composants physiques (par exemple, le processeur, la mémoire, etc.) et exposons les mécanismes d'économie d'énergie que fournissent ses composants. La consommation d'énergie d'un système est directement liée aux caractéristiques de la charge de travail à effectuer, c'est pourquoi, nous présentons et discutons également des différents types de charge de travail (par exemple les charges de travail commercial, scientifique). Puis, nous présentons les approches courantes d'économie d'énergie dans les systèmes distribués en s'intéressant principalement aux *grappes* et aux *nœuds* les constituant. Enfin, nous terminons notre étude en présentant les défis restant à résoudre ainsi que le projet *EcoGrappe*. Nous détaillons ces objectifs et présentons des travaux fait à EDF, partenaire dans ce projet.

Mots-clés : Économie d'énergie, Gestion de ressources, *Green computing*, *Cloud computing*, Virtualisation, Adaption dynamique

1 Introduction

Much work has been done in order to conserve energy on mobile devices as these devices usually do not have a permanent connection to the power grid and solely rely on the limited battery charge. This trend from mobile devices has been disregarded by the high-performance computing (HPC) community for the last few decades. Thus, the main focus mainly relied on improving the performance metrics (i.e. speed) as the only metric of matter at any cost. This has led them to build systems composed out of a huge number of highly power consuming components which quest towards the highest FLOPS (floating-point operations per second) ratio.

A good example for this race is reflected in the TOP500 List (www.top500.org) which ranks the worlds fastest large-scale systems according to the FLOPS metric, extracted from the LINPACK benchmark [14]. Here the performance of the fastest machine between 1993 - 2009 has grown from 59.7 Gflops to 1.75 Pflops which is a 29,5-fold increase.

As the growing performance was mainly achieved by deploying more power consuming components the energy costs of powering those systems have increased significantly. In fact, the Japanese earth simulator which has been leading the TOP500 List between 2002 - 2004 and replaced in 2009 required approximately 18 megawatt of power to achieve its peak performance of 35.86 Tflops, which has resulted in 10 million dollar/year of costs alone for powering and cooling [21]. Other studies have further estimated that power equipment, cooling equipment and energy costs together can make up a fraction of 63% of the total costs of ownership (TCO) of a data center [7]. These data centers alone have consumed 61 billion kWh of U.S. energy in 2006. This is enough energy to power 5,8 million average U.S. households [12]. Newer machines such as the Jaguar Cray XT5-HE have reduced the power consumption but are still measured to draw dizzying 6.9 MW according to the TOP500 List of November 2009.

Besides the huge energy costs, heat dissipation increases inevitably with higher power consumption and doubles the probability of hardware failures [21]. Therefore, reducing the power dissipation has a significant effect on the overall availability, reliability and productivity of a system.

Not least, the way energy is generated influences our environment either directly by the carbon footprint or indirectly by the nuclear waste. Therefore, reducing the energy consumption does not only save a significant amount of money and improves the system reliability, but also helps protecting our environment in a time where global warming approaches towards critical constraints. First attempts to build a power-efficient, reliable and high-performance super-computer can be dated back to 2002. This system was called Green Destiny and composed out of 240 low-power Transmeta processors [61]. Despite all the concerns regarding the performance of this system, it was still able to reach a reasonable peak performance of 101 Gflops at the LINPACK benchmark with a peak power consumption of just 3.2 kW and space requirements of five square feet only. Furthermore, low-power requirements made this system extremely reliable, without any unscheduled downtime during its two years of existence [54]. Since 2004, this trend towards power-efficient computing has been led by the IBM BlueGene/P and BladeCenter QS22 machines which all make use of

the System-on-Chip (SoC) technology to decrease the power consumption and still gain a reasonable performance.

In order to quantify the performance-efficiency of the systems a new metric called FLOPS/Watt was introduced, which serves as the basis measure for the 2007 founded Green500 List (www.green500.org), to rank the most power-efficient systems. After a year of its existence, the first Green500 report [22] from 2009 stated an approximate 75% increase in the average power-efficiency. These results confirm that the HPC community has finally realized the need of energy-efficient computing and thus has started to build more and more energy-efficient systems.

This documents surveys the current energy conservation efforts for commercial clusters and scientific HPC systems. It is organized as follows. Chapter 2 describes the basic principles of power and energy management. Further, it details the workload characteristics of mobile devices (e.g. laptops), commercial web clusters and scientific HPC systems. In the following chapter 3 we survey the current power and energy conservation techniques for distributed systems. Chapter 4 discusses other important issues on distributed systems. Chapter 5 gives a brief introduction to the ANR-founded EcoGrappe project, discusses its objectives and summarizes the results from the work done at our

2 Basic principles of power and energy management

This chapter gives a brief introduction into the basic principles of power and energy management. In particular, we first describe the difference between the two basic terms: energy management and power management. Afterwards we present the current techniques for accounting the power consumption of servers and individual components. Measuring the power consumption is essential in order to understand the system power behaviour and take power management actions.

After describing the available power accounting techniques, we present the most power consuming server resources (i.e. processor, memory, disk, network and power supply) and their low-power modes. These power-modes can be used by the energy management policies to conserve energy.

In order to better understand the server usage we conclude this chapter with a description of the different types of workloads (i.e. mobile, commercial and scientific). This workload knowledge is necessary, as power consumption is directly connected to the type of workload imposed on the system. Further, a deep understanding of workloads is required in order to select the appropriate energy management policy.

2.1 Difference between power and energy management

To follow this document we first define two basic terms: power and energy management, which are used in the entire literature in the area of energy conservation management for distributed systems. Unfortunately, the meaning of these two semantically different terms is not always clear to people outside of this subject and might lead to the wrong assumption of these two terms being equivalent.

In the context of server clusters, power management is referred as the ability to regulate the power consumption of the servers at any discreet point in time. Therefore, it is possible to define a power consumption threshold and make the servers not to exceed this threshold by regulating their power consumption with the low-power modes of the individual server components (e.g. processor, memory, disk, network, etc.). This ability to limit the power consumption allows data center providers to deploy more servers at a given power budget without risking to exceed the air condition capabilities during periods of high load. In addition, it prevents the power grid from being overloaded by consuming less electricity during periods of peak load.

The idea behind energy management is to reduce the total power consumed over a period of time. Thereby, energy management primary aims at reducing the total energy costs and the overall heat dissipation. This is usually done, either by designing low-power hardware or deploying software frameworks which make use of low-power modes, energy-aware job scheduling algorithms, server consolidation or other techniques aiming at reducing the energy consumption during the run-time.

Note that while the primary target of power management is not to conserve energy, as it is only active during short periods of peak load, it is still possible to save a small amount of energy with it during high load. Certainly, the main target of our work is the energy management.

2.2 Measuring the power consumption

One of the fundamental questions when it comes to energy conservation is related to the measurement of the current power consumption. Measuring the power consumption is one of the major steps towards designing energy conservation mechanisms as it is important to identify where it is possible to achieve the most power savings. Thus, we will discuss some of the possible power measurement techniques in this chapter.

The first basic approach to acquire the power consumption of a certain component is to derive the power requirements from the components data sheet. There is a caveat here though, as usually these data sheets only provide information regarding the designed peak power consumption of a component, they neither account the correlation between utilization and power consumption nor the average power consumption.

Another possibility for getting the power consumed on new server generations (e.g. PowerEdge R610) is to use the Intelligent Platform Management Interface (IPMI) [29] which provides a uniform way to access the power-monitoring sensors available on the recent servers. This interface is completely independent of the operating system and thus can be accessed despite of operating system failures and without the need of the servers to be powered on (i.e. connection to the power grid is enough). Further, intelligent power distribution units (PDUs), traditional power meters (e.g. Watts Up Pro power meter) and ACPI enabled power supplies can be used to measure the power consumption of the whole server.

Measuring the power consumption by component (i.e. processor, memory, disk, network, fans, etc.) is a non-trivial task as components usually do not provide any internal power measurement equipment. Thus, in order to measure the power consumption, components need to be isolated. One way to achieve

this is to separate the DC lines providing the energy to the components, integrate a precision resistor in each line and measure the voltage difference at the resistor with a digital meter [27]. This voltage difference can be then used to calculate the current (i.e. voltage/resistance) and thus the power consumption using the $P = Current \times Voltage$ equation. Anyhow, sometimes it is still not possible to separate all the components as some of them share the same power plane (e.g. multicore processors).

Another approach of measuring the processor power consumption, is to model the processor and use this model for simulating the power dissipation under different workloads. Unfortunately, modeling the processor power dissipation is a complex task, as it is heavily dependent on the processors layout and circuit styles. Further, even if all the necessary information is available, running a simulation is usually very time consuming and prone to errors [55]. Hence, alternative approaches are needed to measure the processors power consumption faster and more reliably. Most of the modern processors include hardware performance monitoring units (HPMUs) that can be used to account various low-level operations or events (e.g. TLB misses, cache hits, etc.). This information is exported by the operating system and can be used in order to estimate the processor and even the full system energy consumption [55].

2.3 Power consumption and energy efficiency of resources

Our work aims at conserving energy in clusters. Therefore, it is important to understand the power consumption and the mechanisms available on the individual cluster resources to save energy. This section describes the five most power consuming components (i.e. processor, memory, disk, network and power supply) of a server and gives a brief introduction into the available power management mechanisms. We observe that energy conservation is either done by turning off idle resources or slowing down resources when the demand permits it (i.e. doing work more efficiently). Anyhow, even if the former approach can yield power savings up to 60% depending on the workload intensity [52], performance can be sacrificed during the reactivation time of the resources as resources will be unable to respond to incoming requests until they settle down. The second approach refines the former switch-off strategy and aims at achieving a performance/power tradeoffs without degrading the performance significantly using low-power states.

Given these facts it is obvious that the main target of any energy management policy must be to conserve energy without a significant performance loss because any performance loss implies an increased execution time, which according to the energy equation (i.e. $E = Power \times Time$) results in a proportional energy increase. Thus, any energy management policy which conserves energy and increases the execution time too much can ruin any savings.

Moreover, as mentioned above efficiency of policies which switch off or move resources to lower-power states highly depends on the workload characteristics, since often a performance penalty must be accepted during the time needed for the resources to settle down. Besides the time penalty which might be significant depending on the resource (e.g. node, disk etc.), enormous amount of energy can be wasted due to excessive power state changes (e.g. turn-on and off) [51]. Not least, frequent power state changes can have a bad impact on the reliability of the hardware. Therefore, energy management policies must take into account

the energy overhead and reliability aspects of any power state transitions. We will present different energy conservation techniques for individual server components in this section and show various approaches to use these techniques in the context of distributed systems in section 3.

2.3.1 Processing Unit

The processing unit is the most power-hungry component of a server. Studies have shown that it can draw a fraction from 38% to 48% of the total system power, depending on the imposed workload [23]. Therefore, processor is a good source for power and energy reduction.

Several methods exist in order to conserve power of the processor. As the first naive approach towards conserving power it is possible to select a processor which was initially designed for a mobile device. Such processors usually consume less power as they are designed for devices which are short in battery life. Unfortunately, these low-power processors deliver a lower performance which is often not acceptable for server workloads. Another approach works by selecting a processor according to the best frequency to power ratio. This is often not feasible either as two processors with the same frequency do not necessarily have the same performance (i.e. instructions per cycle) because of different layouts (e.g. use of pipelining).

Alternatively, today's processors no matter either they are designed for mobile devices or servers can either be put into some of the available power states (e.g. HALT) or reduce the voltage/frequency settings in order to save power. In the former approach, power states with different functionality, delay and power consumption tradeoffs can be used to conserve energy. Thus, for example transitioning the processor into the highest power state would result in significant power savings but the longest time in order to return to the normal state, resulting in a delay during which no useful work can be done. Further, every interrupt makes the processor return to the normal state, as interrupt service routines are called to handle the interrupts. If done too frequently this behaviour can lead to enormous transition overheads because switching between different power states consumes additional energy. Therefore, processors need to stay idle relatively long in order to amortize the transitions costs and conserve energy.

A more promising approach makes use of the available performance-states to lower or increase the frequency/voltage depending on the current performance requirements. This technology is called Dynamic Voltage Scaling (DVS) and available in all the recent processors. In order to understand why this method is able to conserve energy we first need to understand the power consumption of a CMOS circuit which is given by the following equation [38] for a particular frequency f :

$$P_f = C \times f \times V^2 \tag{1}$$

where C is the switching capacitance, f the switching frequency and V the supply voltage. According to this equation, it is possible to save power by reducing the frequency. Unfortunately, every reduction in frequency increases the task completion time. Given that application is CPU bound, this time can be approximated appropriately with the following equation:

$$t = \frac{1}{f} \quad (2)$$

That is, time is inverse proportional to frequency. Substituting (1) and (2) into the energy equation $E_f = P_f \times t$ yields to:

$$E_f = C \times V^2 \quad (3)$$

which has no dependency on frequency at all. In this case, reducing the frequency alone is useless as it does not yield any energy savings, slows down the processor and can make other components consume more energy, as they must stay active longer to finish a task. Still, assuming that voltage can be proportionally decreased with frequency combined with the assumption given in equation (2) significant savings are possible especially for applications which can tolerate some delay. The following equation demonstrates it by decreasing the frequency to $\frac{f}{2}$:

$$E_{\frac{f}{2}} = 1/4 \times C \times V^2 \quad (4)$$

Unfortunately, decreasing the voltage has a negative effect on the circuit propagation delay which is defined by the following equation [38]

$$\tau(V) = \frac{k \times V}{(V - V_T)^2} \quad (5)$$

with k constant, V the supply voltage, V_T the threshold voltage ($0 < V_T < V$). We can observe that decreasing voltage/frequency saves power according to (1) and energy according to (4) but increases the propagation delay according to (5). Therefore, it is important to note that frequency needs to be always modified with increased propagation delay in order to ensure proper processor operation. Hence, every reduction in voltage is inevitably leading to lower performance making tasks which rely heavily on the processor longer to complete. Thus, to save energy it is necessary to know the future demands of the processor utilization in order to do a proper voltage/frequency setting. Several DVS algorithms exists to accomplish this job and we will discuss them in the chapter 3.

2.3.2 Main Memory

Main memory is one of the major components of a server system. Organized in banks of hundreds of modules it can consume a significant part of the total server energy (i.e. up to 60%) [15].

Several low-power modes have been introduced by the DRAM manufacturers in order to provide the basis for potential energy savings. In the following discussion we will detail one concrete example based on the Rambus DRAM (RDRAM) specification. Please note that similar power saving techniques exist for the more recent DDR3 SDRAM memory which work by lowering the clock enable (CKE) input.

Each RDRAM module can be transitioned into one of the four available power-states: *active*, *standby*, *nap* and *power-down*. All these power-states differ in the amount of device logic which is switched on (e.g. row/column decoders, refresh circuit, etc.). In addition, each power-state has a different trade-off in

terms of power consumption and delay needed in order to transition back to the active mode. Active state has the highest power consumption, but the lowest data access latency as RDRAM can only read and write data in this state. Lower power-states can be entered when no data needs to be served. Standby has the lowest transition delay but still requires 60% of the active mode power [18]. More power can be conserved by entering the nap mode. This mode consumes just 10% of the active power at the cost of an additional transition delay. Power-down state has the lowest power consumption, but the highest transition delay. These transitions consume additional energy and delay the execution as no data can be processed until the chip has settled down. In particular, the lower the power-state the more time and energy is needed in order to transition the chip into a state in which data can be accessed [44].

Power management algorithms can use these modes to conserve energy when the demand permits it. Thus, it is possible to transition the memory into one of the low-power modes when it is lightly loaded or not used at all. Nevertheless, any transition between the power-modes consumes additional energy and increases the respond times. That is, the ultimate goal of every energy conservation algorithm is to find the best suitable time for power-mode switching depending on the imposed workload. Understanding this, main memory energy conservation has similar energy-time tradeoffs as processors (see above) and disks (see below).

Two types of memory energy conservation algorithms exist: static and dynamic. Static algorithm transition the memory to a predefined power-state (e.g. nap) as soon as there are no pending requests available. Dynamic algorithms transition the memory into the next lower power-state after a predefined time threshold has been reached. Further, different thresholds exist for different power-states. In case of the arrival of a new request the memory is transitioned into the active power-state and waits again until a predefined time threshold has been reached in order to transition back to the next lower power-state. Studies have shown that dynamic algorithms are always better in terms of energy conservation than static algorithms [44].

2.3.3 Disk-based Storage

Most of the today's servers use disk-based storage devices which are made of multiple circular disks called platters. These platters are covered by a thin layer of magnetic material and rotated by a spindle motor. The data is read and written by head which manipulates the magnetization of the material under it. This head is usually placed on a special arm which is used to place the head on the right position.

Energy consumption of disks can make up a big fraction of the total server energy consumption as disks are usually organized in arrays in order to provide fast and redundant storage. Further, the motor spinning and arm movements tend to consume a big fraction of the total disk energy consumption (i.e. 65% depending on the device) [34].

Several methods exists in order to save energy on disks. First approach to save energy is to use the provided low-power modes. In the active mode the disk is doing some work (i.e. seeking, reading, and writing). In the idle mode the motor is spinning but no seeking, reading or writing is done. The standby mode stops the motor, parks the heads and leaves the electric interface on in order

to accept commands. In sleep mode the electronic interface is disabled, no disk operations are accepted anymore and all the data in the disk cache is erased. Only some logic is left powered on in order to accept a reset signal which will put the device back into standby mode. Many devices do not implement this mode and treat it the same manner as the standby mode.

During the regular reading or writing operation the drive is in the active mode and falls down to idle mode as soon as there is no data to be processed. Special software must be used in order to transfer the drive into a lower power-mode (e.g. standby). Power management policies can be implemented in order to use these power-modes to conserve energy. Even though transitioning the drive into a mode which stops the motor (e.g. standby) saves a lot of energy, time and extra energy is needed in order to transition the devices back to a mode where data can be read (e.g. idle). Further, requests can not be handled during the settling time resulting in a delay. Thus, the goal of every energy management strategy is to place the disk into a lower power-mode only when the overhead to transition the disk back to full speed is lower when keeping the disk on, else the energy consumption even increases. Further, frequent transitions are not only bad for the energy consumption but also decrease the reliability of the device.

Another way to conserve energy is to use multi-speed disks. Such devices can be seen as analogous to DVS based processors. That is, they can do work more efficiently when the demand permits it and lower the platters rotation speed [56]. Further, it is possible to make use of the high redundancy and data access popularity of servers in order to conserve energy. Servers are usually over provisioned in order to satisfy high peak demands, deliver high throughput and guarantee data availability (e.g. RAID5). It has been shown that data access patterns tend to follow the Zipf's law. Hence, if we rank the data according to the access frequency we will see that the access frequency of the data is inverse proportional to its rank. That is, some data is more popular than other. This data popularity pattern can be used to concentrate popular data on a subset of disks and shutdown or move other disks to lower-power modes (e.g. standby) in order to save energy [51].

Not least, depending on the future price and technological improvements flash memory based solid state disks (SSDs) could be used to conserve energy in servers. Such devices provide non-volatile storage of data without a need of any mechanical parts (e.g. motor, arm, etc.). Thus, they consume significantly less energy and outperform traditional disks in terms of access performance as no seeks are needed anymore. Still, several limitations need to be resolved before these devices can replace traditional disks. In order to overwrite a segment it has to be erased first. These segments are usually 64-128 KB large. Considering that erasing one byte usually takes around a dozen of micro-seconds, write-performance is relatively poor. Further, segments can be erased and overwritten around 10,000-100,000 times without degrading the performance [46]. Thus, it is the job of the operating system to make sure that segments are not overwritten too often. Finally, SSDs are still around 20 times more expensive as traditional disks and limited in the available capacity.

We can conclude that SSDs are able to conserve a significant part of total energy consumption assuming that software will handle the poor write-performance and the costs per megabyte improve.

2.3.4 Networking

Conserving energy drawn by the networking subsystem can be either done at the node level (i.e. network interface card) or at the infrastructure level (i.e. switches and routers). Energy saving at the node level are barely noticeable as the network interface card is already a very low-power consuming component, composed out of a small amount of memory and a simple processing unit. That is, transitioning the network interface card into a lower power-state does not yield any noticeable savings compared to leaving the card in the active mode [33]. Still, small energy savings are possible when operating at a lower link rate. This technique is called Adaptive Link Rate (ALR) and is similar to the previously introduced DVS method for processing units. Here, the idea is to lower the Ethernet link rate (e.g. transition from 1Gbps to 100Mbps) during periods of low utilization and thus save energy [31].

While energy savings at the node level are relatively small, more savings are possible on the infrastructure level. With the growing demand for bandwidth, routers and switches have started to provide capacity for up to hundreds of terabits. This performance is mainly achieved by improving the speed of the routers switch fabric unit (SFU). This unit is the building block of a router and is in charge of currying the input data to the corresponding output. Consequently, routers have started to consume a non-negligible fraction of the total power consumption. Several efforts have been recently made in order to conserve energy on these devices. Most of these efforts exploit idle times and low-utilization of Ethernet links to turn them off [32] and route the traffic around them.

Not least, further energy savings are possible at the hardware level of the routers. That is, choosing and optimizing the layout of various internal router components (i.e. buffers, links, etc.) is a critical design constraint towards building energy-efficient networking equipment.

We have detailed the current energy-conservation approaches for the networking subsystem in this chapter, as this subsystem is an essential part of a distributed system. As our primary goal is to conserve energy used by the computing equipment (i.e. cluster) we will not further detail the work done in the networking area and refer to the cited documents for further details.

2.3.5 Power Supply Unit

The task of every power supply unit (PSU) is to feed the server resources with power. This is done by converting the high-voltage alternating current (AC) from the power grid to a low-voltage direct current (DC) which most of the electric circuits (e.g. computers) require. In order to switch from higher AC voltage (e.g. 220V) to lower DC voltage (e.g. 12V) additional circuits are required inside the PSU. These circuits convert the voltage and inevitably lose some energy in the form of heat, which then needs to be dissipated by additional fans inside PSU. In order to compare different PSUs it is important to quantify how much energy is lost during the energy transformation. Therefore, the metric DC/AC is used.

If the server components consume 400 Watt and the PSU draws 500 Watt from the power grid it is 80% efficient. The other 20% (100W) are lost in form of heat during the energy transformation. Further, energy efficiency of a PSU mainly depends on the load imposed on the PSU, number of circuits and

other conditions (e.g. temperature). Hence, a PSU which is labeled to be 80% efficient is not necessarily that efficient for all power loads. For example, low power loads tend to be the most energy inefficient ones. Thus, a PSU can be just 60% efficient at 20% of power load.

Studies have further estimated that PSUs are probably the most energy inefficient components in today's data centers as many servers are still shipped with low quality 60 to 70 percent efficient power supplies [8]. One possible solution for energy conservation would be to replace all PSUs by ENERGY STAR certified ones. This certificate is given out PSUs which guarantee a minimum 80% efficiency at any power load.

2.4 Understanding different types of workloads

We distinguish between three types of workloads: mobile, commercial and scientific. While mobile and commercial workloads are interactive, scientific workloads are non-interactive. The interactivity from mobile and commercial workloads comes from the users which are in charge for generating the load. While a mobile device (e.g. laptops) is usually used by a single person at a time, the load generation is limited by a single user. However, a single user usually does not stress all the subsystems (e.g. processing, memory, disk and network) at the same time. Given these circumstances a mobile device usually experiences a lot of idles times which can be exploited in order to deploy power management techniques. Thus, it is possible to slow down the processor (e.g. while reading a document), spin down the disk or just lower the screen brightness without sacrificing the user experience significantly.

Although mobile and commercial workloads are both interactive, commercial workload is usually generated by multiple users. Thus, all the server subsystems can be stressed at the same time, resulting in very little and sometimes even no idles times at all, as the load is usually balanced evenly across the servers (e.g. multi-tier web servers) in order to optimize the resource utilization, increase the throughput, decrease the response times and avoid the servers from being overloaded. This lack of idle times makes the traditional power management mechanisms less applicable to commercial clusters, as transitioning components into lower power states during peak periods of load would result in a significant performance degradation which is often not desired. Furthermore, given the bursty workload characteristics and thus short idle times, components hardly stay idle for a longer period in order to be able to use traditional power-management techniques efficiently (i.e. frequent power mode transitions can have a bad impact on total energy consumption and system reliability) [7]. Since commercial servers are usually over-provisioned in order to sustain peak loads and provide high availability they often use redundant resources (e.g. nodes), big quantity of components (i.e. processors, memories, disks) and inefficient power supplies which dissipate energy. Hence, in order to save energy, new approaches are necessary to be able to make use of traditional power-management techniques. Under utilization, high redundancy, relaxed time constraints and predictability of the workloads (e.g. web services) can be exploited in order to either create or extend idle times for intelligent resource scheduling policies to take energy conservation actions such as shutting down unused nodes (see chapter 3 for more details).

In contrast to mobile and commercial workloads which vary with time, scientific high-performance workloads are non-interactive parallel workloads which communicate according to some internal pattern and exhibit various inter-process dependencies. Usually they are submitted through batch systems, executed on the available resources and the results are processed by the user after they become available [28]. While energy-aware scheduling is used in commercial workloads to adjust the system resources to the imposed load, scientific high-performance workloads are made to speedup a computation. Therefore, computations are usually distributed across the largest number of available nodes, each running at the maximum available speed. Given that time to finish a computation is critical and idle times are rare in HPC, it is often not feasible to shutdown resources as the time overhead to switch them on would slow down a computation. Alternative approaches are needed for scientific workloads to conserve energy and keep the desired performance [25]. These approaches must exploit different workload characteristics in order to save energy. That is, running the servers on the maximum available speed does not necessary lead to a significant performance improvement, as many scientific applications do not have the processor as the major bottleneck resource. Mechanisms which are able to leverage these characteristics can conserve energy with just a limited impact on the performance.

We have summarized the state of the art energy conservation techniques for commercial and scientific workloads in the following chapter 3.

3 Methods for energy consumption management in distributed systems

During the last decade more and more efforts have been put in order to conserve energy on distributed systems. In this section we will survey some of this work done for commercial web clusters and scientific HPC systems. We will first introduce two general energy conservation approaches referred as: low-power and energy-aware computing. After discussing both of them we will concentrate our study on energy-aware computing and describe the currently available state of the art energy conservation policies and algorithms. We divide the work done on energy-aware computing into two categories: node-level and cluster-level energy management. In the first part we will discuss some algorithms which leverage the previously introduced DVS technology in the context mobile devices, commercial web servers and scientific HPC systems. Further, we will introduce other node-level mechanisms such as: Intel Turbo Boost, Core on/off, Request Batching and other related work. Finally, we will finish this chapter with a study of the current cluster-level energy conservation efforts. In particular, coordinated dynamic voltage scaling, turning nodes on/off and virtualization/server consolidation.

3.1 Approaches for energy management

Two general approaches exists in order to conserve energy on distributed systems: low-power and energy-aware computing. The former approach concentrates on improving the performance and energy efficiency by using a large number of low-power processors (e.g. PowerPC 450). The use of low-power

processors makes such systems highly reliable and compact as low-power components generate less heat and can be packed more densely. This approach has evolved from the observation that physical limitations have stopped the frequency improvements of individual processors. Thus, one way to still achieve a reasonable performance and conserve energy is to deploy many low-power processors and write highly parallel software which can make use of them. That is, lower performance of individual processors is compensated by the amount and the parallelism of the software. Still writing a parallel software is a non-trivial and expensive task. Further such systems usually use special architectures (e.g. RiSC) which are expensive and not able to run traditional software without significant modifications. Examples of such low-power computing machines include Green Destiny [21] and IBM Blue Gene/P.

Another approach to conserve energy is called energy-aware computing. The idea behind energy-aware computing is to use less but traditional components in order to achieve energy savings without significant performance degradation. Therefore, it aims at building energy-conscious software frameworks which implement energy-saving policies on top of common hardware. These policies aim at conserving energy by the use of low-power modes provided by the individual server components (e.g. processor, memory, disk, network, etc). Thus, this approach has his own advantages. Energy-aware frameworks work on top of common hardware which is cheap. Further, it is still possible to run traditional software without doing any modifications and save energy.

As our target is not to build a low-power supercomputer, we will not further detail the low-power computing approach and concentrate our state of the art work on energy conservation methods exploiting the ideas of energy-aware frameworks.

3.2 Node level

A lot of work has been done to conserve energy on the node level. Most of this work leverages the research work done in the context of mobile devices during the last two decades. Therefore, in this section we will first introduce some of the well known DVS based algorithms used on mobile devices and distributed systems in order to conserve energy without significant performance degradation. Further, we will detail a new technology called *Intel Turbo Boost* which is sometimes misunderstood and thought to save energy. Finally, we will present other energy conservation methods such as core on/off, request batching and other related work used in the context of mobile devices and servers.

3.2.1 Dynamic processor voltage scaling

One promising approach to conserve energy used by the processor is to do work more efficiently. That is, to lower the voltage/frequency of the processor during low-utilization phases (see chapter 2). Several algorithms have been designed to accomplish this job in the context of mobile devices and distributed systems. In this section we will first introduce some of the common DVS based algorithms found in mobile devices. Afterwards, we will present the recent DVS based work in the context of commercial web clusters and scientific HPC systems.

In order for DVS to simultaneously save energy and preserve the performance it is necessary to set a proper workload dependent frequency. Several algorithms have been designed to find the best trade-off between lower energy and lower delay [62] for **mobile devices** during the last two decades. They can be classified in interval-based and task-based algorithms.

Interval-based algorithms divide the execution time in intervals and make use of the CPU utilization information from past intervals in order to find the optimal CPU frequency for the upcoming interval. These algorithms are simple, work transparently to the application and can be easily implemented as they neither require any application specific knowledge nor any application modifications. Observe, that interval-based algorithms assume that utilization in the future interval will be similar to the one in the past. Given that execution intervals are usually in the range of hundreds of milliseconds such algorithms tend to react slowly to abrupt performance changes and thus can increase the execution time and waste energy for certain workloads.

Task-based algorithms on the other hand side are more efficient than interval-based algorithms as they are able to provide a better energy vs. performance trade-off. Unlike interval-based algorithms these algorithms can not be applied transparently as they require additional application specific knowledge (i.e. deadlines) and existing workload information of each task in order to find the optimal performance level (i.e. voltage/frequency setting). Further, the implementation complexity of such algorithms is high and the required future knowledge (i.e. deadlines) is often very hard to define [53]. Therefore, we will not further discuss these algorithms and concentrate our work on describing some of the well known interval-based algorithms.

Every interval-based algorithm is comprised out of two steps. The first step towards setting the right performance level is to predict the future workload. This prediction is made by taking into account the past processor utilization. Therefore, time is divided into intervals of several dozen milliseconds. Each time the interval starts (i.e. DVS algorithm gets executed), the utilization during the previous interval(s) is determined. This information is then used in the second step to set a proper performance level. A lot of research has been done in the design and implementation of interval-based DVS algorithms. Thus, many algorithms have been developed in the past. We will present some of the well known algorithms referred as: *PAST*, *PEAK* and *AVGn* in the following discussion [58].

PAST: This algorithm predicts that the utilization during the upcoming interval will be the same as during the past interval. That is, it relies solely on the utilization information from the past interval. If the past intervals utilization was high the speed is increased. On the other hand, if the utilization was low the speed is decreased. The amount by which the speed is allowed to be increased/decreased at the beginning of each interval is limited in order to avoid excessive speed variations.

PEAK: This algorithm expects the workload to consist of narrow peaks. It uses the expression of *run_percents* as a ratio between run cycles and idle cycles of the past intervals and predicts a rising *run_percents* to fall down symmetrically and falling *run_percents* to keep falling. Further, a *run_percents* normalized to 1 is assumed to fall whereas a *run_percents*

of 0 to be steady. We can illustrate this behaviour as follows. Define P_{t-1} and P_t as the *run_percents* from the previous two intervals. We can then predict the *run_percents* of the upcoming interval P_{t+1} by using the following notations [30]:

1. If $P_t > P_{t-1}$, then $P_{t+1} = \max\{P_{t-1}, 0.1\}$
2. If $P_t < P_{t-1}$, then $P_{t+1} = \min\{P_t, 0.1\}$
3. If $P_t = P_{t-1}$, then, if $P_t = 1 \rightarrow P_{t+1} = 0.4$ else $P_{t+1} = P_t$

After the prediction, speed setting is done to a value which is fast enough to complete the expected workload.

AVGn: This algorithms makes use of the exponential moving average in order to predict the future behaviour. Thus, its utilization predictions for the upcoming interval are based on the history utilization's of the past intervals. In order to be able to give adjust the importance of older and more recent utilization values the number n must be specified. The equation for calculating the utilization of the upcoming interval is defined as follows [58]:

$$W_t = \frac{nW_{t-1} + U_{t-1}}{n + 1} \quad (6)$$

with W_{t-1} to be me exponential moving average of the past $t - 1$ intervals and U_{t-1} the measured utilization during the last interval. Varying the number n either gives more weight to the past intervals or to the last interval. Thus, a greater value for n favours older values and smaller value newer ones. In case when n equals 0 the utilization of the upcoming interval is considered to be the same as during the last interval.

The authors in [58] have evaluated the described algorithms with various workloads (e.g. MPEG encoding, Compiling, etc.) and stated that the two best algorithms according to the energy savings and performance were PAST and AVGn. Still, as we have already mentioned before assuming that utilization of upcoming interval will be similar to the previous ones is dangerous and can lead to significant performance degradations and unnecessary energy dissipation. More information can be found in [53].

In the context of **commercial web clusters**, several energy conservation policies based on the DVS method have been described and evaluated using a simulator in [16]. This work has evaluated one node level and several cluster level energy conservation policies. For now, we will restrict us to the results from the evaluation of the node level policy and introduce the cluster level policies in the following section 3.3. The introduced node level policy is called *independent voltage scaling (IVS)*. In this policy the performance level of the processor is set independently according to the imposed workload. Thus, each cluster node operates at a different performance level. Still, as the load is usually evenly distributed across the servers using fronted load balancers, all the nodes will operate at approximately the same performance level. Such a basic policy has shown to conserve approximately 29% of energy per node for the given web

workload. At this point it is important to note that this policy is equivalent to the traditional DVS based approaches known from mobile devices (i.e. same interval-based algorithms are used).

Given a **scientific HPC system** it is often not feasible to change the performance level in the same matter as it is done with traditional interval-based DVS algorithms as performance could be scarified significantly. However, scientific workloads often used in the HPC community do not have the CPU as their bottleneck resource. In fact, memory and communication are the most limiting factors for these kind of workloads. Hence, it is possible to save energy without a significant performance degradation if the boundedness of the application can be detected. For example, assumed that some part of the application is memory bound it is possible to reduce the frequency during the execution of this part and save energy.

Some work has been done in order to detect these phases and set the frequency accordingly. In [24], the application is profiled and divided into *phases*. Further, each CPU is assumed to provide multiple energy/performance settings defined as *gear*. Thus, given an application with n phases and g gears, g^n possible gear configurations exists with each configuration being an assignment of a gear to each phase. The authors present a heuristic to search the configuration space of solutions, execute a solution and evaluate its energy vs. time trade-off. In order to evaluate a solution a user-defined metric such as energy-delay product (EDP) is used. Finally, the heuristic terminates as soon as the user (e.g. cluster administrator) specified sufficiency level has been reached. This results in a vector of *gears* which minimize the user-defined metric. The experimental results with the NAS benchmark suite from [24] have shown 9% less energy-usage compared to traditional interval-based algorithms and 16% less when no DVS was used.

Similar work can be found in [36] where the authors detect CPU-boundedness by the use of profiling and information (e.g. cache misses) derived from the performance monitoring units (PMUs) and [26] where the authors evaluate the energy-time tradeoffs in MPI applications.

3.2.2 Intel Turbo Boost

With the introduction of new processors such as Intel Xeon Nehalem-EX having 8 cores, more power-states become available. Additionally, new technologies are getting integrated into the processing units in order to better utilize the cores. Unfortunately, these technologies often get mixed, creating the illusion that traditional DVS based approaches are moved into the processor. One example of such technology is called “Turbo Boost” [37] which is included in the recent Nehalem micro-architecture based Intel CPUs. The ultimate goal of this technology is to boost the processors performance while operating in the highest performance-state (i.e. P0), when the basic conditions such as power consumption, temperature and current draw permit it. That is, this approach can be seen as a method to overclock the processor in a controlled way. Thus, it is possible to run the processor beyond the base operating frequency for some workloads and increase the performance. In order to do this a dedicated power control unit (PCU) was integrated into the processor. This unit adjusts the

voltage/frequency of the processor by monitoring the processors power consumption, temperature and current draw within a closed feedback-control loop. Thereby, given that the constraints are not exceeded the frequency of all the cores can be either increased or decreased in 133 MHz steps when they are active. In the case when only one core is active its frequency can be modified in 266 MHz steps. It is important to note that enabling the “Turbo Boost” technology in the BIOS would typically make the processor cores operate between the base frequency and the “Turbo Boost” maximum frequency. Further, this technique does not yield to any energy savings and even increases the energy consumption. In order to save energy, traditional DVS based algorithms are still implemented at the operating system level using lower power-state (i.e. P1 and later).

3.2.3 Turn cores on and off

Because of the growing number of individual processor cores it is becoming more and more important to manage the individual cores in order to conserve the maximum amount of energy during periods of low utilization. With the introduction of the new Intel Nehalem microarchitecture based CPUs and its power control unit it is now possible for the operating system to shutdown individual cores when the demand permits it. Therefore, these processors provide an extended set of power saving modes know as C-states. This extended set of power saving modes includes a new Deep Power Down state known as C6. In order for a core to enter this power state its cache is flushed and the core state is saved into the shared last level cache. Further, power gates are used to completely shutdown the core in order to avoid wasting energy caused by leakage current. The transitioning between the power states is managed by the *cpuidle* infrastructure within the Linux kernel [49]. This generic infrastructure implements various governors and is in charge for changing the power states of the processor, analogous to the *cpufreq* infrastructure for managing the different performance states (i.e. voltage/frequency levels). These governors decide on the best power state to enter depending on the CPU activity and requirements.

Linux integrates a energy-aware scheduling policy which is in charge for packing the workload such that some cores/packages become idle and thus can be transitioned into a lower power-state by the selected *cpuidle* governor. This policy has shown to perform well for some light workloads which are not memory/cache bound. For memory/cache bound workloads the shared resources such as the last level cache of the cores have become the bottleneck resource. Thus, the performance has suffered significantly as the CPU had to access the main memory more frequently as the result of increased cache misses. Further, this policy does not perform well with short running jobs such as daemons, as these jobs usually finish before the load balancing is invoked and thus remain undetected causing energy wasting wake-ups, reducing the idle times of the cores. Additional interrupts caused by the I/O devices and timers used inside device drivers and application tend to cause unnecessary CPU wake-ups and waste energy. Therefore, interrupt and timer migration is used in Linux next to workload consolidation to maximize CPU sleep time [57]. Further research in this area needs to be investigated.

3.2.4 Request batching

While the previously introduced DVS based policies are well suited for moderate workloads, they still waste energy at low intense workloads because the CPU needs to remain active in order to serve transient workload. Further, there are little energy savings possible for high intense workloads as the CPU needs to run at the highest performance level in order to satisfy the response requirements. Given that web servers exploit a relatively large fractions of idle time, alternative approaches are needed in order to conserve the maximum amount of energy. Here, the basic idea to shutdown idle servers is not always desirable as it would impose additional wake-up and energy overheads. Thus, another policy called “Request batching” was initially introduced in [17] and evaluated with a web server simulator. The idea behind this policy is to accumulate the incoming requests in memory by the network card processor and transition the native processor into low-power state until a predefined per-packet threshold has been reached. Thereby, it is possible to save a significant part of the energy for light workloads as the processor can be moved into a Deep Power Down state (see the paragraph above) avoiding the energy wasting during idle periods.

It is obvious that energy savings and the response times of this policy solely depend on the predefined batching timeout. That is, when the batching timeout is increased more energy can be saved at the cost of increased response times. In order to keep the system responsible the batching timeout needs to be adjusted depending on the imposed workload. In [17], this is done using a closed-loop feedback control. When there are no requests in the queue the processor is transitioned into the Deep Power Down state. The control-loop monitors the system activity (i.e. response times) and adjust the length of the batching timeout accordingly to the predefined steps of 10-100ms. In case the system response time is lower then the predefined system response time target the batching timeout is increased. Analogue if the system response time falls behind the predefined target the batching timeout is decreased. It is important to note that frequent processor power-state transitions consume additional energy and introduce additional time overhead. Hence, request batching is disabled after a certain lower-bound on batching timeout has been reached and enabled when this lower-bound is surpassed. Once disabled, the previously introduced DVS technology can be used in order to achieve additional energy savings for moderate intense workloads. Thus, these two approaches complement each other and can achieve significant energy savings for a wide range of workloads. According to the simulations in [17] CPU energy savings in the range of 17% to 42% are possible in case when the two strategies are used together.

Some work has been done in order to deploy request batching in the context of mobile devices and desktop PCs. Studies have shown that 67% of the desktop PCs stay idle outside the regular work hours. Thus, significant energy savings are possible by transitioning these devices into a lower-power state while still keeping the system responsible for some applications. In [5], a system called *Somniloguy* is introduced which does exactly this. The idea behind this system is to extend the network card by a second low-power microprocessor which is able to serve requests even when the host system is transitioned to a lower-power state. In the case when the host initiates a sleep request the network state of the host is moved to the second microprocessor. This microprocessor runs its own operating system, detects the incoming requests and wakes up the host if

necessary. The complete process is transparent to the remote hosts. Still, even the simplest request initiates a wake-up of the host system, reducing the possible energy savings. In order to increase the energy savings it is necessary for the host system to stay idle as long as possible. Thus, simple tasks such as keeping the *online* status of the IM (e.g. ICQ) can be offloaded to the second processor, increasing the idle time of the host. Therefore, the application developer needs to add a small portion of application specific code, referred to *stub* on the host and the second processor (for more details see the reference). The authors of [5] have validated their work with a prototype implementation and have shown significant energy savings ranging from 60% to 80% for desktop PCs which are usually left on outside of work hours.

3.2.5 Other related work

In the previous chapter we have introduced several DVS based algorithms which are used in a wide range of today's mobile devices and servers in order to conserve energy. All these algorithms assume that the CPU is the major power consuming component of the system. Unfortunately, this is not always the case especially in low-power computing systems such as mobile devices or IBM BlueGenes. Here the energy consumption of the memory can equal the one from the processing unit. Given such a scenario traditional DVS based algorithms fail in achieving significant energy savings. One approach to solve this problem is to combine DVS based algorithms with power-aware memory. In [19], the authors made initial efforts in studying this interaction and show that significant energy savings up to 89% are possible depending on the workload.

Another way to conserve energy and provide a certain level of QoS is to use transcoding. The initial idea behind transcoding relies in the need of web services to provide low-latency to its customers. Providing low-latency access to the web service is ultimately constrained by the networking bandwidth of the provider and the customer. Thus, techniques are needed in order to preserve the low-latency, serve as many clients as possible and still provide a certain amount of QoS. In [11], the authors show that transcoding allows the server to manage its bandwidth efficiently in the sense that it does not introduce additional latency or limits the service usage. Further, clients can be divided into classes and get different allocations of bandwidth and service quality. This does not only help to manage the bandwidth efficiently but also saves energy as processing the data (e.g. images) of lower quality can be accomplished with lower performance (i.e. CPU utilization).

3.3 Cluster level

In the previous section we have introduced some of the recent node level energy conservation work. Further, we have discovered the DVS method to be one of the most promising techniques in the context of mobile devices and servers to conserve a significant amount of energy with respect to the given workload. However, even though node level approaches no matter at what layer they are used (i.e. hardware or software) have shown to conserve a significant fraction of the total energy consumption, idle power consumption of a server is still relatively high (e.g. 190W for IBM eServer 325 [48]). Thus, in order to increase the energy savings, existing node level approaches can be complemented by higher

level energy-aware frameworks. Such frameworks can address the problem of high idle power consumption either by consolidating individual tasks or virtual machines on a subset of nodes and switching off the unused ones. In this section we will present some of the recent work done in this area, in particular on coordinated dynamic voltage scaling, node on/off and virtualization technologies.

3.3.1 Coordinated Dynamic Voltage Scaling

One of the proposed policies in [16] is called *coordinated voltage scaling (CVS)*. This policy requires all the nodes of a cluster to have DVS-enabled processors. Additionally, one node needs to be selected which runs a monitoring service. All the nodes inform the monitoring service periodically about their current frequencies. This information is used to calculate the average frequency setting for the whole cluster, which is then propagated to all the nodes. The receiving nodes then set their performance level to the calculated setting. As the calculated average frequency does not necessary exactly match the processor supported discreet performance levels, the processors are usually set to a frequency around the calculated one. This policy slightly (1%) outperforms the previously introduced IVS policy, as running the cluster nodes at a certain frequency F is more efficient then running the nodes at independent frequencies those average is F . Still the implementation complexity and the low energy savings compared to the IVS method make this policy less applicable within an productive environment.

3.3.2 Turn nodes on and off

One of the main objectives of our work is to design energy management policies which are able to reconfigure the cluster according to the imposed workload and turn off the unused servers. Turning off unused servers is advantageous as there is little difference in power consumption between idle and fully utilized servers (i.e. ca. 25% only). Thus, consolidating the workload on a subset of servers and turning off the unused ones always saves power. Some research has been done in this area, most of it in the context of **commercial web clusters**. In [50], first steps towards building a reconfigurable cluster of web servers have been made. The authors have proposed a load concentration algorithm and implemented it on a cluster-based web server and at the OS level using the Nomad cluster operating system. Their algorithm assume a homogeneous cluster setup and a stateless system (e.g. web server). Hence, it is possible to remove nodes without the need to migrate any tasks. In the context of the cluster operating system they rely on the provided migration functionality. The proposed load concentration algorithm works by recording the resource demands (e.g. cpu, disk, etc) periodically and based on this information it calculates when it is possible to turn on/off a node. Therefore, a so called degradation limit is specified. This limit specifies how much performance degradation is acceptable when turning off a node and consolidating its load to another server. Suppose we have a cluster of two nodes. Each node has a demand for disk of 70% and 60% respectively. If we add this two values we get 130% demand for disk. Thus, this demand could be served by one node, in case a performance degradation of at least 30% is acceptable. In this particular scenario the algorithm decides to remove one node. Other use cases are possible with larger configurations where the deci-

sion could be either to turn on additional nodes when the degradation limit is exceeded or turn off single or multiple nodes with either no degradation or the one specified within the degradation limit. If the decision is to add another node, it is necessary to determine which load should be sent to this node. It is obvious that load should be moved away from heavily loaded machines. In case of a node removal its necessary to determine which machine needs to be turned off and where to send its current load. Here it is important to select nodes to be removed which are undergoing low utilization and destination nodes where additional load would not cause the system to get overloaded.

The authors in [50] do not present any details on the node selection and migration decisions. During their evaluation of the modified web server no migration is needed. In case of a removal decision they simply pickup the node with the lowest utilization. When a node addition is needed no load migration is necessary either as it is managed by the web server load balancer automatically. In the context of the operating system, load needs to be migrated. Thus, after the decision to remove a node has been taken, the algorithm selects two node as source/destination with the lowest utilization of the resources and migrates all the load from the source node to the destination node. Possible unbalance caused by such a migration is assumed to be managed by the OS load balancing policy. Further, as only one node addition at a time is allowed it might take a while until the system is able to react to increasing resource demands. The authors have evaluated their implementation under moderate increasing workload without any performance spikes. Under such a workload they could reduce the power consumption of the web server by 86% and simultaneously conserve 43% of the energy compared to a static 8 node setup. Similar results were reached for the OS with 86% for power consumption and 32% for energy while fixing the maximum performance degradation at 20%.

This vary-on/vary-off policy was further refined in [16] and combined with the previously introduced coordinated DVS policy in order to achieve further energy savings. The authors have used a validated simulator in order to evaluate their new policy by using web workloads. Savings up to 48% by a simple node on/off policy were reported. Additional energy savings (i.e. up to 18% more) were achieved by combining this policy with the previously introduced coordinated dynamic voltage scaling policy. The newly introduced policy assumes that the power consumption of the server is dominated by its processing unit. Therefore, the processing unit power consumption model is applied in order to model the system power consumption. As introduced in chapter 2, dynamic power consumption of the CPU is given by the following equation:

$$P(f) = C \times f \times V^2 \quad (7)$$

Further in DVS based processors voltage needs to be reduced in proportion to the frequency. Hence, voltage can be expressed as linear function in frequency:

$$V = \lambda \times f \quad (8)$$

with lambda being the proportionality factor. Substituting into the equation (7) and accumulating the constants yields into the following frequency dependent equation:

$$P(f) = c_1 \times f^3 \quad (9)$$

Assumed that the power consumption of all other components (e.g. disk, network, etc.) of the system is constant following model can be constructed for the server power consumption:

$$SP(f) = c_0 + c_1 \times f^3 \quad (10)$$

where c_0 includes the base power consumption of the processing unit and the power consumption of the other components of the system. Assuming a cluster of n homogeneous servers its power consumption at a particular frequency f can be approximated by the following equation:

$$n \times SP(f) = n \times (c_0 + c_1 \times f^3) \quad (11)$$

Given the equation of the cluster power consumption it is now possible to precompute a table a frequencies f_{off} and f_{on} for any specified number n of servers and use these frequencies as thresholds in order to either turn on or turn off the servers depending on the current cluster-wide frequency.

In particular this works as follows. When the cluster-wide frequency f is low it is possible to save energy by turning off a server and redistribute the workload on the other $n - 1$ servers. Turning off one server is beneficial as there is little difference between the power consumed of a machine which is idle and fully utilized. Thus, turning off a server saves power with just a small amount of energy increase on the other servers. This energy increase is caused by the consequential increase in frequency to $\frac{n \times f}{n-1}$ on the other servers in order to preserve the performance of the system.

In order to find a frequency at which $n - 1$ servers consume less energy when n servers we need to solve the following inequation for f :

$$(n - 1) \times SP\left(\frac{n \times f}{n - 1}\right) < n \times SP(f) \Leftrightarrow f < \sqrt[3]{\frac{c_0}{c_1} \frac{(n - 1)^2}{2n^2 - n}} \quad (12)$$

This indicates that as soon as the frequency falls beyond $\sqrt[3]{\frac{c_0}{c_1} \frac{(n-1)^2}{2n^2-n}}$ it is beneficial to turn off one node. Assumed that the constants c_0 and c_1 are given it is now possible to precompute the turn off frequencies for any number n of servers. Analogous it can be beneficial to turn on additional server during periods of high load in order to be able to reduce the overall frequency of the cluster. Reducing the overall frequency conserves energy as frequency is proportional to voltage and energy consumed by the processor has a quadratic dependency on voltage (see chapter 2). Thus, reducing the voltage has a significant effect on the total energy consumption. When the new server is turned on the workload can be distributed across $n + 1$ servers leading to a lower overall utilization. Thus, frequency of the servers can be lowered to $\frac{n \times f}{n+1}$. Consequently in order to find a frequency at which $n + 1$ servers consume less energy when n servers following inequation needs to be solved for f :

$$(n + 1) \times SP\left(\frac{n \times f}{n + 1}\right) < n \times SP(f) \Leftrightarrow f > \sqrt[3]{\frac{c_0}{c_1} \frac{(n + 1)^2}{2n^2 + n}} \quad (13)$$

Thus, as soon as the cluster frequency exceeds $\sqrt[3]{\frac{c_0}{c_1} \frac{(n+1)^2}{2n^2+n}}$ it is beneficial to turn on one more server and decrease the overall frequency of the cluster.

This approach was used in [16] in order to precompute a frequency table for $1 \leq n \leq 20$ servers and run a simulation of a clustered web server. More details regarding this approach can be derived from the cited document.

The previously described approaches have worked well in the context of web servers. Here the idea was to use a frontend load balancer to concentrate the incoming client requests on a viewer number of servers in order to create idle times and shutdown the unused ones. This has worked well because of the nature of web server workloads. These are short-term independent client requests which can be scheduled independently. Further, such systems usually do not hold any internal state making it a straightforward decision to shutdown unused nodes. Hence, the efficiency of this approach is mainly dependent on the ability of the system to properly predict the upcoming workload in order to know when it is appropriate to shutdown the nodes or to switch them on again in order to sustain the load. Unfortunately, many workload do not follow the patterns of web workloads. For example numerical computations are often coarse-grained and need to maintain some internal state. Thus, it is not possible to turn off the nodes without taking additional actions such as individual task migration. Some work has been done in order to provide energy savings under these types of workloads at the cluster resource manager level.

In [39], a resource manager called SLURM (Simple Linux Utility for Resource Management) is proposed. SLURM offers a power saving mechanism which transitions nodes into a lower-power state (e.g. power down) when a predefined idle-period has been reached. Similarly, Sun Grid Engine (SGE) [3] in its latest release supports power saving features [4] which involve powering off idle nodes.

Furthermore, in [13] an energy-aware framework for experimental grids (i.e. Grid5000 [10]) is proposed. The idea behind this framework is to exploit the concept of reservations in order to conserve energy. Thus, when a reservation is issued by a user it is associated with several user specified parameters such as as: number of required nodes, duration and estimated start-time. This information is recorded in the resource manager agenda and used by the scheduler in order to manage the resources efficiently. The proposed framework extends the existing resource manager of Grid5000 called OAR [9] by a prediction module which makes use of the resource manager reservation agenda in order to predict when it is appropriate to turn on/off a server. Further, it can aggregate existing reservations depending on the user specified policy in order to reduce frequent node turn off and on actions. Thus, a user can either specify a policy which will fit the users demands (i.e. start the reservation on the specified time) or one which will try to aggregate the reservations in order to minimize the amount of servers to boot/shutdown and thus save energy. The authors show that in case when the latter policy is used energy savings up to 44% are possible. This work is currently being finalized and will be available in the future OAR release [1]

Finally, in the category of commercial resource managers, *Moab* [41] and *PBS Pro* [2] offer several energy-saving mechanisms which involve turning off machines when the average load falls below a predefined threshold.

3.3.3 Virtualization

Server virtualization and consolidation are two complementing approaches which are able to offer significant energy savings. Here, the idea is to virtualize the servers by running the tasks within individual virtual machines and exploit the

existing live migration functionality of today's hyper-visors in order to consolidate the VMs on a fewer number of servers and turn off the unused ones during periods of low workload [45]. Achieving such a consolidation is challenging as consolidating the VMs on a smaller number of servers still needs to fulfill the resource requirements of the VMs. Thus, it is necessary to find a node configuration such that the resource requirements of the VMs are satisfied and the amount of servers needed is minimized. In order to find such a node configuration it is necessary to know the available resources (e.g. cpu, memory, disk, network, etc) of each node and the resource requirements of the VMs. Finding the optimal configuration when becomes an instance of the NP-hard bin-packing problem [40], which is unsolvable for problems of large size. Therefore, a heuristic solution is needed in order to achieve reasonable results.

Some research has been done in order to address the problem of finding the most power efficient VM placement. This work has resulted in simulations and implementations of several VM placement policies. In [45], a scheme called Power-Aware Domain Distribution (PADD) is introduced and verified using a simulation. The idea behind this scheme is to find a VM distribution such that the energy consumption is minimized and the performance requirements are met. The authors have evaluated their schema under various utilization rates (7.5% - 14.5%) and were able to achieve energy savings up to 92.5% of their idealized system, which always has the lowest energy consumption without any performance degradation. Other work aims at implementing a prototype system. In [35], a consolidation framework named *Entropy* is presented. This framework makes use of the *Constraint Programming (CP)* paradigm in order to find the optimal node configuration for a particular set of VMs. Such a configuration is said to be optimal as it uses the smallest possible number of nodes to host the VMs. Further, the authors show that consolidation overhead is determined by two criteria: time to find the optimal node configuration and time to migrate the VMs to the particular configuration. Unfortunately, no impact of the consolidation on energy consumption has been presented in this study.

Similar more recent efforts can be found in [59],[43], [47], [20] and [6]. Here, the VM placement problem is usually modeled as one-dimensional or multi-dimensional bin-packing problem and a *heuristic algorithm* is used to minimize the number of machines hosting the workload.

Energy conservation by the use of virtualization and consolidation remains to be a hot research topic with a lot of work currently going on. We refer to the cited documents for more information.

4 Other important issues on distributed systems

The previous chapter has introduced some of the energy conservation work done in the context of mobile devices, commercial web clusters and scientific HPC systems. We have observed that most of these work solely concentrate on providing energy-management solutions for homogeneous systems. Further, we have concentrated our work on techniques which primary target on finding the best energy vs. performance trade-off. Nevertheless, power consumption of today's data centers keeps growing, rising the need for solutions which are able to cap the power needs of the servers. This is necessary in order to avoid

power grid breakdowns and server hardware failures due to insufficient cooling capabilities.

In this chapter we will introduce the two aspects and provide some references to the ongoing work in these areas.

4.1 Heterogeneity of servers and clusters

Most of the today's productive clusters are heterogen. Thus, cluster nodes most differ in their performance, capacity and power/energy consumption. While the initially installed hardware components get older they are getting replaced with more power-full ones in order to meet the growing resource demands. Moreover, hardware components which have experienced some errors are replaced by new more power-full ones, as the cost-to-performance ratio keeps falling [34]. In fact, data centers tend to replace the state of the art PC-like nodes by chassis of "blade" nodes. These chassis have a centralized power, networking and cooling equipment with higher server density and lower space requirements. Furthermore, most of the "blade" nodes make use of low-power components (e.g. 2.5-inch disks, modified PowerPC CPUs, etc.) allowing them to be more energy efficient than traditional PC-like nodes. Thus, given such a configuration clusters are homogen at most when they were initially installed. Given such an environment, it is a non-trivial task to find a cluster configuration which achieves the best trade-off between power and performance for a given workload. More research needs to be investigated in this area as most of the current approaches assume a homogeneous setup.

4.2 Managing peak power consumption

The growing demand for computing power leads the data centers towards deploying increasing numbers of high-density servers (e.g. blades). These servers use less power compared to traditional 1U servers and ease the management of the infrastructure as they do not require much cabling overhead. Still aggregated together they can draw a lot of power and require enormous cooling because of the tight packing within the chassis. A common approach to resolve this problem is to over-provision the power and cooling supply systems. This might be beneficial for some workloads but adds additional costs to the data center. Thus, the idea behind peak power consumption management is manage the power consumption dynamically depending on the current system requirements. When the system runs below its power capacity its processor frequency is set to the maximum value, otherwise the frequency is decreased until the power capacity constraints are satisfied again. This approach results in more power efficient systems as they are designed to handle real workloads and still are able to operate under rare unexpected workloads with just a small performance degradation. Further, this technique allows the data center operators to match the power consumed by the servers to the available cooling capabilities and increase the number of servers in case of some performance degradation is tolerable. In [42], a peak power management solution at the system-architecture level is introduced. The authors use a closed-loop feedback controller within the service processor firmware in order to monitor the current power consumption of the whole system, and take actions such as reducing the CPU frequency if

the power constraints are violated. The feedback controller proposed consists of three parts: monitor, controller and the actuator. Each time the controller is executed the current power is measured and passed as input to the controller. The controller then calculates the new processor frequency setting and passes it to the actuator, which is in charge for adjusting the frequency. The authors show that closed-loop feedback controllers outperform traditional industrial solutions such as: ad-hoc and open-loop controllers as they are able to provide a very accurate control of the system power consumption. Other work concerning using feedback control theory in peak power consumption management can be found in [60], where the authors use a MIMO (Multiple Input Multiple Output) controller in order to provide better performance and a more accurate control. Still, most of these works concentrate solely on the adjusting the processor frequency and do not take into account the power consumption of the other components (e.g. disks, memory, etc). Taking into account the power consumption of these components is necessary in order to provide a more accurate control system. Further, processor intensive workloads can profit from low power needs of other components as it is possible to re-distribute the power budget and thus improve the performance. More work needs to be done in this area.

5 EcoGrappe - A new initiative towards building energy-aware cluster system

The growing demand for computing power has led the high-performance computing (HPC) community to build systems composed of a huge number of power consuming components during the last few decades. Therefore, the associated energy costs for powering and cooling those systems have suddenly started to become a critical concern for data center operators. Hence, several efforts such as Green Destiny, BlueGene/L, The Green500 List, The Green Grid, INRIA GREEN-NET and COST Action IC0804 have been initiated in order to address the problem of the increasing energy consumption.

EcoGrappe is a new project started in the beginning of 2009 by the three partners: *INIRA Rennes, Kerlabs and EDF*. It is partially funded by the French National Research Agency (ANR) and complements the existing energy conservation initiatives. The ultimate goal of the EcoGrappe project is to combine and extend the existing energy conservation efforts within a energy-aware framework for distributed computing cluster. This framework will implement various energy-aware scheduling policies such as task consolidation and node addition/removal in order to conserve energy when the demand permits it. Further, Kerrighed operating system and its customizable global scheduler will serve as the basis, as they provides the required functionality in order to verify our work (i.e. migration and node addition/removal).

In this chapter, we will give a brief introduction into our objectives and present the results from the work done at our project partner Électricité de France (EDF).

5.1 Objectives

Much work has been done in order to conserve energy on mobile devices and commercial web clusters. In the previous chapters we have detailed some of this work. We have noticed that even though significant energy savings are possible at the node level, idle power consumption of the servers is still significantly high. Thus, in order to eliminate the enormous idle power consumption, higher level approaches such as cluster-level energy management frameworks are needed.

Unfortunately, less work has been done in providing such a framework for a distributed computing cluster and in the design of job scheduling policies which take the energy consumption into account. Our work aims at filling exactly this gap.

In a typical distributed computing cluster a scheduler is used in order to optimize the resource utilization of the servers. Hence, it makes use of various scheduling policy such as *round-robin* in order to optimize the cluster utilization. Given that all the servers of a cluster are executing some workload there is less opportunity to save energy. Therefore, alternative energy-aware scheduling policies are needed which are able to redistribute the load in order to create idle times and turn off the unused nodes or take other energy conservation actions. These policies depend on various parameters such as the number of outstanding jobs, job duration, resource requirements, resource availability, resource utilization and power consumption. Assumed that the required parameters are available it is possible to take actions such as:

1. Scheduling jobs on the smallest number of nodes and turn off the unused ones when the load is low.
2. Scheduling jobs according to the energy costs (day/night).
3. Scheduling jobs to the most energy-efficient (FLOPS/Watt) servers within a heterogeneous cluster environment.
4. Learning application specific energy-consumption and scheduling jobs to the server configurations with the best performance vs power trade-off.
5. Provide *Green hints* (e.g. delay execution) to the users and save energy.
6. Avoiding hot spots by moving load away from *hot* servers. Hot servers are machines which are overloaded and do not receive enough cooling.

Our next steps will involve proposing an architecture of our framework, designing new energy-aware task placement policies and validating our work within a prototype implementation of the framework. In the following section we will present our initial steps towards this goal done at our project partner EDF.

5.2 Work done at EDF

Électricité de France (EDF) uses many high-performance computing platforms. These are used in its R&D as well as in its engineering and financial operations. As the electrical consumption of these platforms represents a large fraction of their overall cost of ownership; the objective of EDF in the EcoGrappe project is to reduce their overall energy consumption.

To measure the impact on the energy consumption of the different policies developed in this project, it is necessary to have the tools to evaluate the energy footprint of our codes. These footprints are related to the performance of each of our codes with their own characteristic, taking into account various launching parameters for each code. The energy footprint of an idle machine must also be assessed. These measures must be made on at least one HPC platform, and if possible on several types of platforms with different characteristics (e.g. processors, memory, etc).

The tools developed in the EcoGrappe project need to be able to compare the results obtained:

- On different hardware architectures
- With different execution parameters (e.g. number of nodes)
- Between two successive runs, before and after application of various policies to reduce consumption that will be developed

The differences in resource consumption (e.g. CPU, memory, network, etc.) and energy consumption may be significant, depending on the hardware used, but also according to the distribution of the code on these various resources. The effect of the repartition of the code on to the resources needs to be verified.

EDF R&D currently has an internship on this topic. Omar Bouslama, a student in his final year at the University of Technology of Compiègne (UTC) has been working on this subject since February 2010 and will finish in July 2010. The first step in this study was to install the equipment needed to perform the measurements. Based on our existing HPC test platform, we fitted all of the power distribution equipment with sensors on each 220V socket to measure the instantaneous power consumption. In this manner, the consumption of every server, computing node, piece of networking equipment or storage server is measured individually. As part of this platform consists of nodes equipped with graphics cards, we can also measure the impact of GPU computing on the power consumption for codes that use this architecture. During the installation of the power measurement equipment, temperature and humidity sensors were installed on the front and rear side of each bay of the computer room, to be able to have accurate data on the distribution of hotspots and their evolution over time.

In the course of the internship, we established the following work schedule:

- Brief survey of the state of the art;
- Handling of tools for the measurement of the power consumption, temperature and humidity;
- Development of usable software tools for the collection and use of data;
- Handling of a reference code to establish and refine a measurement methodology. The EDF code “Code_Saturne” was chosen because it is well known and easily portable to different hardware configurations. The “Linpack” benchmark, used in the context of the “TOP500” to measure the raw performance of HPC computers, has also been used.

- Comparative studies of different codes commonly used at EDF (e.g. Code_Saturne, Telemac, Code_Aster, etc.) to extract standard profiles and search the optimum balance between computation time and power consumption, according to the execution parameters chosen. This study is conducted in direct collaboration with the developers of these codes, so that the results are usable in their algorithmic research to reduce energy consumption.
- Proposition of an approach to present the results of the measurement, including a profile of the execution of the code itself.
- Execution of a series of measurements by varying the parameters used when launching the code (e.g. number of nodes, number of cores per node, amount of memory per node, processors frequency, etc.)
- Proposition of ways to improve the instrumentation
- The use of hardware counters implanted in processors by using standard interfaces that have appeared in the latest versions of Linux kernel.
- Implementation of the proposed methodology on other codes, and extension of this methodology on aspects not provided with Code_Saturne.
- Presentation of results.

The results of this internship will be integrated into the EcoGrappe project.

In practise, to measure the performance of an equipment, it is necessary to periodically collect data such as memory consumption and CPU load. This data can be made available by the operating system. However, data concerning the instantaneous power consumption are either not available at the operating system level, unreliable or of a granularity such that little information of the energy consumption of a single process can be derived. Certain manufacturers are incorporating sensors inside their new equipment, and sometimes several sensors per motherboard so that individual elements of the server can have its consumption measured. However every manufacturer uses its own technology and there is currently no standardized tool to collect information from these devices. Pragmatically, we have chosen to collect aggregate consumption of each server, via a sensor connected on its external power supply.

The sensors themselves are connected to an Ethernet network, and they may be interrogated by a dedicated client, using the SNMP protocol. The classical monitoring software using this protocol (e.g. Nagios, Cacti, etc.) cannot be used for energy consumption measurements, as they are incapable of assuring a pseudo “real-time” measurement. As our goal is to have the capacity to measure every second with a precision exceeding a second, we have proposed to develop a specialised set of tools base on SNMP to perform the data collection. These tools are written in the Python language which, with extensive use of classes to maximize the code reuse, while being easy to maintain, with capabilities to easily interface with the operating system, and allowing the storage of retrieved data into a database.

Initially, the data collection system was written using an snmpd daemon running on each node. This method allowed us to have a low impact on the overall performance of the compute nodes. After updating the Linux kernel to version 2.6.32, we will attempt to directly use the performance measurement

tools that are now integrated. The portability of the information gathered from the kernel 2.6.32 does not yet seem to be complete according to the current processors generation and manufacturers. We believe, nevertheless, that we will obtain a better accuracy using this method.

A supervisor is in charge of the data collection process. The supervisor interrogates each node and each equipment, as defined by an XML file used to configure the supervisor. The XML file lists every node it wishes to collect data from, and the file lists the information to collect. Information about the CPU load is collected in the form of processor JIFFIES1 used by a process and requires for further processing to transform this data into a CPU load. It is necessary to calculate the load from different clocks, retaining only the relevant information. The information recorded is stored in a MySQL database. We need to precisely date each measurement and event related to a particular code. Therefore, all servers and measurement equipment are synchronized using the NTP protocol. A relay server provides this function for the cluster.

Installing Code_Saturne: Code_Saturne is an open source computation fluid dynamics code internally developed by EDF R&D. The version used here is based on OpenMPI. The code can then be run on multiple nodes sharing the same file system. For the purposes of experimentation, a NFS distributed file system is implemented on one of the servers of the platform and then mounted on the computing nodes.

The Code_Saturne is installed on the shared partition. Once setup for a simple and commonly used benchmark case (the “FATHER” test case), Code_Saturne is executed multiple times using different configurations of the number of nodes and/or the number of processors per node.

Initially, we ran the code on multiple nodes directly with OpenMPI without using a batch manager or scheduler. We used these results to refine the different modes of measurement. The execution time and the consumption are measured on several devices, so we have to ensure that all our systems logs are well synchronized.

To measure the energy footprint, we must measure the energy consumption on all equipment involved in the computation, and if possible, we must separate the part of energy used for the code itself from the part consumed by the system for its own needs (to include losses related to power supply, ventilation, etc.).

The cluster is fully instrumented, including network equipments, KVM, etc. Initially, we will study only the consumption of compute nodes. We are aware that a significant fraction of the energy is dissipated for common services, shared between all the codes executed on the cluster. We will try to estimate this value. In our study, we will simply determine a coefficient to be applied to the measured value; the material does not allow a fine measure “component by component”.

We will use the tools for measuring performance in a way that is not traditional, so we must also ensure the relevance of what you measure. For example if we want to measure a running time based on the number of system clock cycles, what happens if we adjust the clock frequency? What is the accuracy of our measurements?

The user interface is developed in PHP and based on Ajax to allow continuous update of the display and the jQuery framework to simplify the use of javascript. The visualisation is done via a web interface, portable to any conventional browser. The selection of displayed data is made interactively through menus. The server extracts the raw data from the MySQL database, performs the necessary formatting calculations and transmits the results to the browser in a json data format (similar to XML). The processing of such data for display is then produced by the “flot” library. This allows the graphics to be manipulated zoomed in or expanded or to update the display without reloading the page.

The interface is also designed to compare the results of several simulations and zoom in on a part to observe the details. We have therefore several pages in the interface.

- One to watch the real time evolution of the study for both code execution and power consumption of every server. There may be a latency between the computation and the display of the data, which is not significant for us, so that the measures are well synchronized.
- In another page, we have the tools dedicated to post-project analysis and comparison between the results of several measurement campaigns. These tools will retrieve the values stored in the database. The data are then transmitted to the browser, in which appropriate treatment is applied to achieve the desired display.

A presentation of this work is planned at future RMLL Software Meeting in Bordeaux, at the beginning of July 2010.

6 Conclusion

This document has surveyed the current state of the art in energy management techniques for servers and distributed systems. We have started our document with the introduction into the energy consumption issue of today’s clusters and realized that energy costs for powering and cooling those systems have started to become a primary concern for data center operators. Thus, building energy-aware systems will be necessary in order to reduce the dramatically increasing energy costs and the carbon footprint of the clusters.

During our study of the state of the art we have introduced various techniques for measuring the power consumption of today’s servers, identified major power consuming server components and discussed current ways on how to reduce the power consumption of these components. Various components of the system provide low-power modes which can be used by higher level energy management policies in order to conserve energy when the demand permits it. Thus, components can be transitioned into a lower-power mode depending on the current utilization. However, this must be done carefully as various low-power modes have different tradeoffs in terms of energy and performance. Hence, if transitions are done too frequently energy consumption can be even increased.

After our study of the power-management possibilities of individual hardware components we have focused our work at energy conservation methods for servers and distributed systems and classified them into two categories: low-power computing and energy-aware systems, which is equivalent to hardware

level and software level energy management. While it is important to design energy efficient hardware, software can help in order to exploit low-power modes provided by the hardware and increased the energy savings. We have identified that energy conservation techniques known from mobile devices such as DVS can be adapted to servers on the node level and complemented by alternative approaches such as core on/off, request batching, transcoding and compiler optimization in order to optimize the energy efficiency. Despite the fact that node level approaches no matter where they are implemented (hardware or software) can achieve significant energy savings, higher level (cluster/grid) solutions are needed to eliminate the high idle power consumption of the servers. Thus, scheduling policies which are able to migrate workload and turn off/on whole servers cleverly are necessary. Further, virtualization of servers and consolidation of virtual machines by the use of live migration is an important aspect in today's high density packed data centers and can help to conserve a significant portion of the total energy consumption.

We can conclude that a holistic approach is needed in order to achieve maximal energy savings. Future systems need to be designed in such way that they guarantee to be energy-aware at all the abstraction layers (i.e. hardware, operating system and software). In addition they have to take heterogeneity of components into account and deploy mechanisms in order react to abrupt power requirements changes.

Besides our study of the state of the art in energy management for servers and distributed systems we have introduced a new energy conservation initiative called EcoGrappe, detailed its objectives and presented the results from the work done at our project partner EDF. In our next steps will work on proposing an energy-aware architecture, designing new energy-aware task placement algorithms and verify them within a prototype implementation. More details on this work will be presented in our next deliverable D2.2.

7 Acknowledgment

This research was funded by the french *Agence Nationale de la Recherche (ANR)* project EcoGrappe under the contract number ANR-08-SEGI-000.

References

- [1] OAR Energy Savings: http://oar.imag.fr/works/gsoc/2008/gsoc_energy_saving.html. 3.3.2
- [2] PBS Works: <http://www.pbsworks.com/>. 3.3.2
- [3] Sun Grid Engine (SGE): <http://gridengine.sunsource.net/>. 3.3.2
- [4] Sun Grid Engine (SGE) Power Saving: <http://wiki.gridengine.info/wiki/index.php/PowerSaving>. 3.3.2
- [5] Yuvraj Agarwal, Steve Hodges, Ranveer Chandra, James Scott, Paramvir Bahl, and Rajesh Gupta. Somniloquy: augmenting network interfaces to reduce pc energy usage. In *NSDI'09: Proceedings of the 6th USENIX*

- symposium on Networked systems design and implementation*, pages 365–380, Berkeley, CA, USA, 2009. USENIX Association. [3.2.4](#)
- [6] Anton Beloglazov and Rajkumar Buyya. Energy efficient allocation of virtual machines in cloud data centers. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:577–578, 2010. [3.3.3](#)
- [7] Ricardo Bianchini, , Ricardo Bianchini, and Ram Rajamony. Power and energy management for server systems. *IEEE Computer*, 37:2004, 2003. [1](#), [2.4](#)
- [8] Chris Calwell. What lies within: Improving the efficiency of internal power supplies, 2008. [2.3.5](#)
- [9] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, and O. Richard. A batch scheduler with high level components. In *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 2 - Volume 02*, CCGRID '05, pages 776–783, Washington, DC, USA, 2005. IEEE Computer Society. [3.3.2](#)
- [10] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, GRID '05, pages 99–106, Washington, DC, USA, 2005. IEEE Computer Society. [3.3.2](#)
- [11] Surendar Chandra, Carla Schlatter Ellis, and Amin Vahdat. Differentiated multimedia web services using quality aware transcoding, 2000. [3.2.5](#)
- [12] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association. [1](#)
- [13] Georges Da-Costa, Jean-Patrick Gelas, Yiannis Georgiou, Laurent Lefèvre, Anne-Cécile Orgerie, Jean-Marc Pierson, Olivier Richard, and Kamal Sharma. The green-net framework: Energy efficiency in large scale distributed systems. In *HPPAC 2009 : High Performance Power Aware Computing Workshop in conjunction with IPDPS 2009*, Rome, Italy, May 2009. [3.3.2](#)
- [14] Jack Dongarra, Piotr Luszczek, and Antoine Petit. The linpack benchmark: past, present and future. *Concurrency and Computation: Practice and Experience*, 15(9):803–820, 2003. [1](#)
- [15] Matthias Eiblmaier, Rukun Mao, and Xiaorui Wang. Power management for main memory with access latency control. 2009. [2.3.2](#)
- [16] E.N. (Mootaz) Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy-efficient server clusters. In *In Proceedings of the 2nd Workshop on Power-Aware Computing Systems*, pages 179–196, 2002. [3.2.1](#), [3.3.1](#), [3.3.2](#), [3.3.2](#)

- [17] Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy conservation policies for web servers. In *In Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, 2003. [3.2.4](#)
- [18] Xiaobo Fan, Carla Ellis, and Alvin Lebeck. Memory controller policies for dram power management. In *ISLPED '01: Proceedings of the 2001 international symposium on Low power electronics and design*, pages 129–134, New York, NY, USA, 2001. ACM. [2.3.2](#)
- [19] Xiaobo Fan, Carla Schlatter Ellis, and Alvin R. Lebeck. The synergy between power-aware memory systems and processor voltage scaling. In *PACS*, pages 164–179, 2003. [3.2.5](#)
- [20] Eugen Feller, Louis Rilling, Christine Morin, Renaud Lottiaux, and Daniel Leprince. Snooze: A Scalable, Fault-Tolerant and Distributed Consolidation Manager for Large-Scale Clusters. In *2010 IEEE/ACM International Conference on Green Computing and Communications (GreenCom-2010)*, Hangzhou, China, December 2010. [3.3.3](#)
- [21] Wu-chun Feng, Xizhou Feng, and Rong Ge. Green supercomputing comes of age. *IT Professional*, 10(1):17–23, 2008. [1](#), [3.1](#)
- [22] Wu-Chun Feng and Tom Scogland. The green500 list: Year one. In *5th IEEE Workshop on High-Performance, Power-Aware Computing (in conjunction with the 23rd International Parallel and Distributed Processing Symposium)*, Rome, Italy, May 2009. [1](#)
- [23] Xizhou Feng, Rong Ge, and Kirk W. Cameron. Power and energy profiling of scientific applications on distributed systems. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 34, Washington, DC, USA, 2005. IEEE Computer Society. [2.3.1](#)
- [24] Vincent W. Freeh and David K. Lowenthal. Using multiple energy gears in mpi programs on a power-scalable cluster. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 164–173, New York, NY, USA, 2005. ACM. [3.2.1](#)
- [25] Vincent W. Freeh, David K. Lowenthal, Feng Pan, Nandini Kappiah, Rob Springer, Barry L. Rountree, and Mark E. Femal. Analyzing the energy-time trade-off in high-performance computing applications. *IEEE Trans. Parallel Distrib. Syst.*, 18(6):835–848, 2007. [2.4](#)
- [26] Vincent W. Freeh, Feng Pan, Nandini Kappiah, David K. Lowenthal, and Rob Springer. Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 4.1, Washington, DC, USA, 2005. IEEE Computer Society. [3.2.1](#)
- [27] R Ge, X Feng, S Song, H Chang, D Li, and K Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1 –1, 2010. [2.2](#)

- [28] Rong Ge, Xizhou Feng, and Kirk W. Cameron. Improvement of power-performance efficiency for high-end computing. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*, page 233.2, Washington, DC, USA, 2005. IEEE Computer Society. [2.4](#)
- [29] Ravi A. Giri. Increasing data center efficiency with server power measurements. 2010. [2.2](#)
- [30] Kinshuk Govil and Edwin Chan. Comparing algorithms for dynamic speed-setting of a low-power cpu, 1995. [3.2.1](#)
- [31] Chamara Gunaratne, Kenneth Christensen, Bruce Nordman, and Stephen Suen. Reducing the energy consumption of ethernet with adaptive link rate (alr). *IEEE Trans. Comput.*, 57(4):448–461, 2008. [2.3.4](#)
- [32] M. Gupta and S. Singh. Dynamic ethernet link shutdown for energy conservation on ethernet links. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 6156–6161, June 2007. [2.3.4](#)
- [33] Maruti Gupta, Satyajit Grover, and Suresh Singh. A feasibility study for power management in lan switches. In *ICNP '04: Proceedings of the 12th IEEE International Conference on Network Protocols*, pages 361–371, Washington, DC, USA, 2004. IEEE Computer Society. [2.3.4](#)
- [34] Taliver Heath, Bruno Diniz, Enrique V. Carrera, Wagner Meira Jr., and Ricardo Bianchini. Energy conservation in heterogeneous server clusters. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 186–195, New York, NY, USA, 2005. ACM. [2.3.3](#), [4.1](#)
- [35] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 41–50, New York, NY, USA, 2009. ACM. [3.3.3](#)
- [36] Chung hsing Hsu and Wu chun Feng. Effective dynamic voltage scaling through cpu-boundedness detection. In *In Workshop on Power Aware Computing Systems*, pages 135–149, 2004. [3.2.1](#)
- [37] intel2008. Intel turbo boost technology in intel core microarchitecture (nehalem) based processors, November 2008. [3.2.2](#)
- [38] Tohru Ishihara and Hiroto Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED '98: Proceedings of the 1998 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 1998. ACM. [2.3.1](#), [2.3.1](#)
- [39] Morris A. Jette, Andy B. Yoo, and Mark Grondona. Slurm: Simple linux utility for resource management. In *In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*, pages 44–60. Springer-Verlag, 2002. [3.3.2](#)

- [40] David S. Johnson, Alan J. Demers, Jeffrey D. Ullman, M. R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, 1974. [3.3.3](#)
- [41] Andre Kerstens and Steven A. DuChene. Applying green computing to clusters and the data center. Linux Symposium 2008, 2008. [3.3.2](#)
- [42] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Server-level power control. In *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*, page 4, Washington, DC, USA, 2007. IEEE Computer Society. [4.2](#)
- [43] Bo Li, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *CLOUD '09: Proceedings of the 2009 IEEE International Conference on Cloud Computing*, pages 17–24, Washington, DC, USA, 2009. IEEE Computer Society. [3.3.3](#)
- [44] Xiaodong Li, Zhenmin Li, Yuanyuan Zhou, and Sarita Adve. Performance directed energy management for main memory and disks. *Trans. Storage*, 1(3):346–380, 2005. [2.3.2](#)
- [45] Min Yeol Lim, Freeman Rawson, Tyler Bletsch, and Vincent W. Freeh. Padd: Power aware domain distribution. *Distributed Computing Systems, International Conference on*, 0:239–247, 2009. [3.3.3](#)
- [46] Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron. Migrating server storage to ssds: analysis of trade-offs. In *EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems*, pages 145–158, New York, NY, USA, 2009. ACM. [2.3.3](#)
- [47] Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 265–278, New York, NY, USA, 2007. ACM. [3.3.3](#)
- [48] Anne-Cécile Orgerie, Laurent Lefèvre, and Jean-Patrick Gelas. Chasing gaps between bursts : Towards energy efficient large scale experimental grids. In *PDCAT 2008 : The Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 381–389, Dunedin, New Zealand, December 2008. [3.3](#)
- [49] Venkatesh Pallipadi. cpuidle - do nothing, efficiently... [3.2.3](#)
- [50] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *In Workshop on Compilers and Operating Systems for Low Power*, 2001. [3.3.2](#)
- [51] Eduardo Souza De Albuquerque Pinheiro. *Energy conservation for server systems*. PhD thesis, New Brunswick, NJ, USA, 2005. Director-Bianchini, Ricardo. [2.3](#), [2.3.3](#)

- [52] Johan Pouwelse, Koen Langendoen, and Henk Sips. Dynamic voltage scaling on a low-power microprocessor. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 251–259, New York, NY, USA, 2001. ACM. [2.3](#)
- [53] Euiseong Seo, Seonyeong Park, Jinsoo Kim, and Joonwon Lee. Tsb: A dvs algorithm with quick response for general purpose operating systems. *Journal of Systems Architecture*, 54(1-2):1–14, 2008. [3.2.1](#), [3.2.1](#)
- [54] Sushant Sharma, Chung hsing Hsu, and Wu chun Feng. Making a case for a green500 list. In *In Proc. of the Workshop on High-Performance, Power-Aware Computing*, 2006. [1](#)
- [55] Karan Singh, Major Bhadauria, and Sally A. McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37(2):46–55, 2009. [2.2](#)
- [56] Minseok Song. Energy-aware data prefetching for multi-speed disks in video servers. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 755–758, New York, NY, USA, 2007. ACM. [2.3.3](#)
- [57] Vaidyanathan Srinivasan, Gautham R. Shenoy, Srivatsa Vaddagiri, Dipankar Sarma, and Venkatesh Pallipadi. Energy-aware task and interrupt management in linux. volume 2, August 2008. [3.2.3](#)
- [58] David Tam, Winnie Tsang, and Catalin Drula. Dynamic voltage scaling in mobile devices, 2003. [3.2.1](#), [3.2.1](#), [3.2.1](#)
- [59] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264, New York, NY, USA, 2008. Springer-Verlag New York, Inc. [3.3.3](#)
- [60] Xiaorui Wang and Ming Chen. Cluster-level feedback power control for performance optimization. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 101–110, Feb. 2008. [4.2](#)
- [61] Michael S. Warren, Eric H. Weigle, and Wu-Chun Feng. High-density computing: a 240-processor beowulf in one cubic meter. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press. [1](#)
- [62] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced cpu energy, 1994. [3.2.1](#)

Contents

1	Introduction	3
2	Basic principles of power and energy management	4
2.1	Difference between power and energy management	4
2.2	Measuring the power consumption	5
2.3	Power consumption and energy efficiency of resources	6
2.3.1	Processing Unit	7
2.3.2	Main Memory	8
2.3.3	Disk-based Storage	9
2.3.4	Networking	11
2.3.5	Power Supply Unit	11
2.4	Understanding different types of workloads	12
3	Methods for energy consumption management in distributed systems	13
3.1	Approaches for energy management	13
3.2	Node level	14
3.2.1	Dynamic processor voltage scaling	14
3.2.2	Intel Turbo Boost	17
3.2.3	Turn cores on and off	18
3.2.4	Request batching	19
3.2.5	Other related work	20
3.3	Cluster level	20
3.3.1	Coordinated Dynamic Voltage Scaling	21
3.3.2	Turn nodes on and off	21
3.3.3	Virtualization	24
4	Other important issues on distributed systems	25
4.1	Heterogeneity of servers and clusters	26
4.2	Managing peak power consumption	26
5	EcoGrappe - A new initiative towards building energy-aware cluster system	27
5.1	Objectives	28
5.2	Work done at EDF	28
6	Conclusion	32
7	Acknowledgment	33



Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399