An Integer Linear Programming Representation for DataCenter Power-Aware Management

Josep Ll. Berral, Ricard Gavaldà, Jordi Torres
Universitat Politècnica de Catalunya and Barcelona Supercomputing Center
berral@ac.upc.edu, gavalda@lsi.upc.edu, torres@ac.upc.edu

November 12, 2010

Abstract

Nowadays energy-related costs have become one of the major economical factors in IT data-centers, and companies and the research community are currently working in new efficient power-aware resource management strategies, also known as "Green IT". But despite power awareness, the classical factors like economical benefit and quality of service are still essentials for the correct performing of the Cloud businesses. So efficiency must meet power saving, economic benefit and client satisfaction, points often all-three incompatible.

This work exposes how to represent a grid data-center based scheduling problem, taking the advantages of the virtualization and consolidation techniques, as a linear integer programming problem including all three mentioned factors. Although being integer linear programming (ILP) a computationally hard problem, specifying correctly its constraints and optimization function can contribute to find integer optimal solutions in relative short time. So ILP solutions can help designers and system managers not only to apply them to schedulers but also to create new heuristics and holistic functions that approximate well to the optimal solutions in a quicker way.

The proposed model is evaluated comparing the optimization results in an only power-aware scheduling, where the system minimizes the power consumption via consolidation; a revenue-based scheduling, where the system maximizes the benefit of running machines against the power consumption; and a quality of service oriented scheduling, where some relaxation towards QoS is tolerated in exchange of avoiding expensive operations. Results demonstrate that ILP is useful in order to find good schedules and the lower bounds for the scheduling optimization, so new heuristics and policies can be obtained by weighting the involved elements (revenue, power, quality of service) and observing the system behavior.

1 Introduction

Nowadays the concept of the Cloud has become a hot topic in the externalization of information and IT resources for people and enterprises. The introduction of

new IT infrastructures and kinds of services into the business world, the Internet population has grown enough to make the need for computing resources an important matter to be handled. The possibility of offering "everything" as a service (platform-as-a-service, infrastructure-as-a-service, services-as-a-service) has allowed that enterprises that had all their IT in private owned data-centers to look for outside hosting and computing companies to trust their data or IT needs. As a consequence, the Cloud and its opportunities have brought lots of computing resources companies to start a *data-center race* offering computing and storage resources at low prices.

These new Cloud data-centers respond to the demands of the people around the world using them, having single web applications easily used by millions of persons, and data to be available from everywhere. I.e. services offered by Google and Youtube must be replicated around the globe, or just be efficient enough to move data, jobs or applications among the *data-center farms* spread along the planet, in order to serve their services with a desired Quality of Service (QoS). So given the amount of applications running now and in the next future on the Cloud, the task of administrating its resources and services in an efficient way becomes a hard optimization problem.

For this optimization there are several factors to keep in mind. Economic factors are basics for an enterprise, as obtaining high revenues are the principal goal. Further bringing an adequate service to the clients trying to grant the maximum client satisfaction is another goal to meet, that often enters in conflict with obtaining the maximum revenue as requires some cost. Finally one of the principal costs and almost important element to keep in mind is the power consumption. Having a powerful data-center in order to satisfy the demand is not enough. Energy-related costs have become a major economical factor for IT infrastructures and data-centers, as this energetic cost is reflected in the electric consumption, growing not necessarily linear with the capacity of that data-centers (powering machines, powering coolers, energy transport, energy production), also with the natural environment [**?**] and the social and government pressures. Enterprises dedicated to data-center services and also the research community is today being challenged to find efficient ways to manage all this growing IT systems keeping in mind not only the "traditional" goals but also power consumption.

Finding the optimal solutions, or the best solution given an adequate computing time, passes through an *intelligent management*. Modeling the Cloud and each involved element, the architects and designers can obtain new ways to understand the system and find algorithms and methods able to manage the Cloud in a more accurate way. Also this management is tending to be automated, as having system operators keeping an eye over the system constantly and reacting in advance to any incident is often not possible. Autonomic Computing is the research field responsible of automating the management of systems by observing them, collecting knowledge, and make decisions towards system goals, letting the systems manage themselves (self-healing, self-protection, self-optimization and self-configuring).

This work presents a data-center modeling focused on the three main goals: economic revenue, power saving and quality of service. This model represents a data-center as a set of elements (machines, processors and jobs) to be scheduled in an optimal way, setting for each element a set of constraints to be accomplished,

representing the capabilities of each one and its requirements.

Thanks to technologies like *virtualization*, data-centers are able to reunite several jobs in physical machines isolating them, sharing resources, and the most important, being able to migrate running the jobs from a machine to another. The proposed scheduling model uses the advantages of this technology in order to dynamically allocate jobs in the optimal places while *consolidating*. Consolidation is a power-saving method consisting in reducing the number of wasted resources by allocating the most number of jobs in the lowest number of machines as possible without compromising things like QoS or performance. A consequence of consolidating is take the most advantage as possible of the physical machines resources, and being able to shut down all those emptied resources. One of the goals of the model is to shut down as many machines as possible by consolidating its jobs in order to avoid powering idle machines.

The proposed methodology models a grid based data-center as a set of resources and as a set of jobs, each one with its energetic or resource requirements, involving a consumption cost and a rewarded execution. The problem to be solved at each scheduling round is to decide what resources are assigned to each job, while maximizing a function result of the benefit obtained by the jobs revenues, the power costs and how much loss of QoS can be tolerated in order to save energy. For this reason, this approach represents the model as an Integer Linear program (as some of the elements cannot be split), solving each scheduling using well known linear programming techniques and integer optimizers.

It is well known that solving an integer linear program can become a hard computational problem, also the integer optimization is often fitted within a time function, but having good constraints and a well defined problem can help to find, in a short time, solutions very close to the lower bounds of the problem. This lets the middleware specialists to understand better the system and be able to create good heuristics close to the optimal/sub-optimal solutions found with ILP solvers. In fact, this work will be really useful in order to obtain data and information towards analyze the behavior of grid data-centers and learn new management rules.

2 Related Work

In this section some related work on the key techniques used in this approach are explained. This key techniques are *consolidation*, *virtualization*, and *turning on and off machines*,

2.1 Virtualization and Consolidation

Virtualization is one of the key technologies in the Cloud, which has enabled cost reduction and easier resource management in service providers. As virtualization allows to run several processes, jobs, guest OS, and also virtual machines in one or several physical machines or platforms, it allows the consolidation of applications, multiplexing them onto physical resources, and supporting isolation from other applications sharing the same physical resource. Tasks can be run everywhere and be

migrated without many handicaps on the base systems, but also virtual machines can perform optimizations over the host OS and physical machine.

This virtualization technology has become a hot research topic for maximizing benefits, but it has added another layer of abstraction to the management systems, preventing or making more complex the conventional energy management for performing efficiently or correctly in virtual environments. During the last past years, works like the ones presented by W. Vogels [17] studied the consolidation advantages using virtualization while other works, like the ones from R. Nathuji et al.[14] have widely explored its advantages from a power efficiency point of view.

Recent works by V. Petrucci et al. [15] proposed a dynamic configuration approach for power optimization in virtualized server clusters and outlines an algorithm to dynamically manage it. All these techniques, applying consolidation policies, are mainly focused on a power efficiency strategy taking into account the cost of turning on and off resources, as it is explained in next subsection. Also VM migration and VM placement optimization is aimed in the works of L. Liu et al. [13], in order to improve the VM placement and consolidate in a better way. Based on these works, I. Goiri et al. introduces the SLA-factor into the self-managing virtualized resource policies [7, 8]. The SLA-driven policies look for facilitating resource management in service providers, allowing the cost reduction and at the same time the SLA agreed QoS fulfillment.

2.2 Turn on-off machines

Another key technique, related with virtualization and consolidation in to determine when to turn on and off physical machines. Works like the presented by I. Goiri, O.Fito et al. [8, 5] show that decrease the number of on-line machines obviously assure a decrease of consumed power, but also the system is often unable to bring service given an increase of load, so a compromise between on-line machines and energy saving must be found. In their works this decision is driven by means of two thresholds: the minimum *Working nodes* threshold λ_{min}, which determines when the provider can start to turn off nodes, and the maximum *Working nodes* threshold λ_{max}, which determines when the provider must turn on new nodes. After modeling specific loads and machine consumptions, using different kinds of scheduling and consolidation techniques, the influence of the turning on/off thresholds by showing the SLA and the power consumption can be evaluated. An adequate thresholds can be obtained (this time empirically) in order to decide how many physical machines are needed on-line, and shut down the rest of them.

Based on the same works, J.Ll. Berral et al. [3] proposes a framework that provides an *intelligent* consolidation methodology using different techniques like the turning on/off machines, power-aware consolidation algorithms, and machine learning techniques to deal with uncertain information while maximizing performance. Using the information from system behaviors, the machine learning approach uses a learned model in order to predict power consumption levels, CPU loads, and SLA timings, and improve scheduling decisions. The experiments realized using Grid workloads and a Cloud environment demonstrate how consolidation aware policies give a better energy efficiency in front of non-consolidating ones, an also the machine

learning model responses much better with respect to power consumption when the information obtained from users and tasks is not uniformly accurate.

This technique is also applied in an approach by I. Kamitsos et al. [12], which sets unused hosts in a low consumption state in order to save energy. In this technique a Bellman's function, based in dynamic programming and recursive methodology, is used to decide when to set into sleeping status those hosts that are not needed maintaining the other submitted jobs into the on-line hosts.

But turning on and off is not limited to machines, also components and resources can be started up and shut down. Policies can decide whether set on or off the full machine or a specific component, and works like the exposed by Y. Tan et al. [16] show a framework for controlling the system power manager using reinforcement learning algorithms. In this case, the learner uses a Q-learning algorithm, a popular reinforcement learning algorithm originally designed to find policies in Markovian Decision Processes.

3 Integer linear programming approach

A grid based data-center can be modeled as a set of resources, each one with a consumption cost, and a set of jobs to be executed with a set of resource requirements and a set of benefits and execution penalties. The problem to solve at each scheduling round is to decide what resources are assigned to each job, depending always in its requirements and conditions established by the agreement between the provider and the client submitting the job (Service Level Agreement or SLA). The best solution will be that one that maximizes or minimizes a target function, usually a function describing the benefit of the solution.

In the proposed situation, the three elements to maximize or minimize are the power consumption, the economic benefit and the client satisfaction, so a good solution is that one assigning resources to jobs (or jobs to resources) saving the most electrical power while granting a good quality of service and having a positive benefit value by serving that clients. So the problem can be represented by a function to maximize the optimal balance between the three elements, and a set of conditions and rules in order to set jobs and resources without overloading the resources and granting viable and real solutions.

At this time we consider that a job can not be split between two or more hosts. When scheduling a job into a host, the job remains completely in that host. this makes the problem to turn into a Integer problem, so the elements conforming the solution must remain integers and a linear real programming solution can not be applied here as a valid solution.

3.1 Scheduling Approach

The solution for the problem defined here is a scheduling integer binary matrix $Hosts \times Jobs$, where each position $[h, j]$ indicates whether the job j is or not in host h. A valid solution must accomplish the condition that a job must be run entirely in a unique host, so it can not be split in different hosts at this time.

Each job needs determined resources to run properly at each moment, like CPU quota, memory space and I/O access. At this time, as an example and case of study we focus on CPU resources, understanding that memory and I/O can also be represented following the same model and expanding it. With this purpose, the system must be able to observe the CPU consumption of each task, and also the available CPU for each host. A solution looking for assuring the jobs requirements must allocate jobs in hosts in a way that each job has its CPU quota, and the host is able to give that required CPU. So the sum of the jobs CPU demands in a given host must not surpass the CPU capabilities.

These conditions are the basic constraints in the problem solving process. So a consistent solution consists in a schedule where jobs are in one host only, the load of a host does not surpass the capabilities of it:

$$
\begin{array}{rl}
\text{Variables:} & schedule[Hosts, Jobs], \text{ as Integer Binary ; representing the Schedule} \\
\text{Parameters:} & cpus(h), \text{ as CPUs existing in host } h \\
 & cons(j), \text{ as the CPU consumption of job } j \\
 & \\
\text{Constraints:} & Unique\{j \in Jobs\} := \sum_{h \in Hosts} schedule[h, j] = 1 \\
 & Capacity\{h \in Hosts\} := \sum_{j \in Jobs} schedule[h, j] \cdot cons(j) \leq cpus(h)
\end{array}
$$

Note: Consider that a job CPU consumption can be a positive real value instead of a positive integer value. When a job enters in CPU keeps the CPU for itself during its time quantum, but as jobs tend to block themselves waiting for I/O or resting in IDLE/sleeping/nice status, given a determined amount of time the expectation of CPU consumption can differ from rounded values.

3.2 Minimizing the power cost

The main goal of this article is to schedule properly data-centers while minimizing the power consumption. At this first stage, the search problem focuses on reducing the number of CPUs used to run the data-center jobs by consolidating them. The idea of consolidation consists in running the commended jobs using the least resources as possible, and in our case it means running as more jobs as possible in as less hosts as possible. As shown in our previous work [8], given a host the power curve grows in a logarithmically alike way, understanding that two machines with many processors running only one each of them consumes more power that only one of those machines running two processors, keeping the second one in a low-power state or just shut down.

In order to model the CPU power consumption of a given host, the CPUs of it can be considered as On-Line or Off-Line depending on the required load. New technologies and architectures not only allow to maintain multi-processor in IDLE states but also shut down processors and components on demand. I.e. a load of 2.50 CPU should make a host to run 3 CPUs, leaving a 0.5 CPU in IDLE state (consuming power) and letting a job with a demand of 0.5 CPU to enter and take

profit of this CPU waste. And when CPU load goes beyond 2.0 the host is allowed to shut down a CPU reducing the consumption waste.

This brings new conditions to be accomplished: the solution will consider that the number of CPUs given a host is a natural value, so the schedule will also consider separately the processors (CPUs) of a given host. Also, the number of active CPUs in a host will not surpass the maximum number of CPUs available on that host (this condition looks silly at first time, but must be considered when having a model with different shaped hosts in the data-center).

In order to keep the representation of the problem as much linear and clear as possible, the value to minimize is the sum of all the power consumed by the scheduled solution and its active CPUs. This is, for each host, look at how much CPUs are running, and check the power consumption according to the host characteristic power curve. I.e. the function to minimize for the host represented, consisting in a 4 CPUs Xeon machine following its power curve, would be $pr_1 \cdot 267.8 + pr_2 \cdot 17.7 + pr_3 \cdot 17.0 + pr_4 \cdot 15.4$ (In this case coefficients are kWatts/hour). Then, the sum of the power consumption of all the data-center is the sum of all power measures. This representation of the problem requires a last condition in order to make it reliable, as CPUs will be started up and shut down in order, so the first CPU to be active in the representation model will be pr_1, then pr_2 and successively.

The resulting model brings as a solution a scheduler minimizing the consumed power taking into account the consolidation goals, like i.e. that filling a half-empty host is better that starting a stopped one:

Variables:	$schedule[Hosts, Jobs]$, as Integer Binary ; representing the Schedule
	$pr_i[Hosts]$, as Integer Binary ; representing the use for each host of its i-essim CPU
Parameters:	pwr_i, as the power consumed by an i-essim CPU
	$cpus(h)$, as CPUs existing in host h
	$cons(j)$, as the CPU consumption of job j
Minimize:	$\sum_{h \in Hosts, i \in cpus(h)} pr_i[h] \cdot pwr_i$
Constraints:	$Procesors\{h \in Hosts, i \in cpus(h)\} := pr_i[h] \geq pr_{i+1}[h]$
	$Unique\{j \in Jobs\} := \sum_{h \in Hosts} schedule[h, j] = 1$
	$MaxCPU\{h \in hosts\} := \sum_{i \in cpus(h)} pr_i[h] \leq cpus(h)$
	$Capacity\{h \in Hosts\} := \sum_{j \in Jobs} schedule[h, j] \cdot cons(j) \leq \sum_{i \in cpus(h)} pr_i[h]$

Note that constraints corresponding to the previous section are modified according to the new introduced variables.

3.3 Maximize economic benefit

Once having modeled system from a power consumption point of view, the economic factor can be introduced. Successfully accomplished jobs are rewarded with

revenue, so clients pay the data-center provider for running their applications on it. Given this obvious fact, jobs can be translated to money according the data-center pricing or Service Level Agreements (SLA). Power consumption can be represented as kWatt/hour, tons of CO2, or also money per kWatt/hour, evaluating the power consumption by the cost of buying the required electricity. In this manner, revenues and power costs can be included in the same equation: $Benefit = Revenue - Costs$.

When jobs and power have a fixed value, and job revenue is above power cost, benefit will always imply running the most applications as possible while consolidating, so power waste is minimized. Unfortunately, consolidation strategies have a great handicap: migration costs. Changing a job from a host to another implies that during this process the job is stopped or replicated, and this can make the SLA fail due to broken deadlines or interruptions of service, and also extra CPU load while moving the job. For this reason, migration is penalized with an economical cost referring to agreements between client-provider or to time and resource wasting [7].

At this time, the model attempts to maximize the benefit of the data-center, as the revenue for all tasks minus the power cost, and minus a penalty for each migration done towards the previous schedule (excluding finished and new-coming jobs). That penalty can be considered, i.e., as the nonpayment for the migration time, an insurance for the unaccomplishment of the SLA, or predict the risk of an SLA failure and its consequences. As an example, we will consider the nonpayment for that migration time, but keeping in mind all the other options:

Variables:	$schedule[Hosts, Jobs]$, as Integer Binary ; representing the Schedule
	$pr_i[Hosts]$, as Integer Binary ; representing the use for each host of its i-essim CPU
Parameters:	pwr_i, as the power consumed by an i-essim CPU
	$cpus(h)$, as CPUs existing in host h
	$cons(j)$, as the CPU consumption of job j
Functions:	$migr(schedule_{old}, schedule)$, as $\frac{1}{2} \cdot (schedule_{old} \oplus schedule)$ (w.o. leaving or new jobs)
Maximize:	$+ \sum_{h \in Hosts, j \in Jobs} schedule[h, j] \cdot$ Revenue Job/Hour
	$- \sum_{h \in Hosts, i \in cpus(h)} (pr_i[h] \cdot pwr_i) \cdot$ power price
	$- migr(schedule_{old}, schedule) \cdot time_{migration} \cdot$ Revenue Job/Hour
Constraints:	$Procesors\{h \in Hosts, i \in cpus(h)\} := pr_i[h] \geq pr_{i+1}[h]$
	$Unique\{j \in Jobs\} := \sum_{h \in Hosts} schedule[h, j] \leq 1$
	$MaxCPU\{h \in hosts\} := \sum_{i \in cpus(h)} pr_i[h] \leq cpus(h)$
	$Capacity\{h \in Hosts\} := \sum_{j \in Jobs} schedule[h, j] \cdot cons(j) \leq \sum_{i \in cpus(h)} pr_i[h]$

Note that the impact of migration can affect CPU loads and many other relevant factors on the system, depending on the relation of the migration time and rescheduling time.

3.4 Quality of Service as a factor

Often systems can be enough flexible to allow some tolerance to Quality of Service, and that means that jobs are not strictly tight to a fixed QoS, and sometimes this QoS can be relaxed letting the system to certain overload in order to improve consolidation and reduce power consumption. By relaxing the QoS, some penalization can be applied specified in the SLA, so the schedule is able to alter the demands of each job by attending at the economic consequences.

In order to define the level of accomplishment of the job goals and SLA conditions, the concept *Health* is defined. The health is an index indicating how well is performing a job, and often depends on the amount of the required resources are received. A value of 1 means that the job is performing optimally, and 0 means that the job is not running or not progressing in its execution.

When turning the jobs CPU requirement into variables, in a range between a maximum CPU (original required) and minimum CPU (SLA failure assured), we can estimate the health level using knowledge from the system, heuristics, or prediction [3] comparing the offered CPU with the maximum required CPU. This health value can be used to establish the penalty to be applied or subtracted to the revenue, or by default use a fixed value explicitly indicated in the SLA.

The function to be optimized includes now the new factor, by scaling the revenue with the health function, adding the before fixed job requirements as variables, and establishing a range for that variable. Changes to be introduced are reflected in the following parts of the problem:

Variables:	$jcpu[Jobs]$, as Integer ; representing the CPU usage of job j
Parameters:	$consmin(j)$, as the minimum required CPU consumption of job j
	$consmax(j)$, as the maximum required CPU consumption of job j
Functions:	$health(j)$, level of QoS of the job j result of $jcpu[j] \sim \langle consmin(j), consmax(j) \rangle$
Constraints:	$MarginCPU\{j \in Jobs\} := 0 < consmin(j) \leq jcpu[j] \leq consmax(j)$
	$Capacity\{h \in Hosts\} := \sum_{j \in Jobs} schedule[h,j] \cdot jcpu[j] \leq \sum_{i \in cpus(h)} pr_i[h]$
Maximize:	$+ \sum_{h \in Hosts, j \in Jobs} (schedule[h,j]) \cdot$ Revenue Job/Hour
	$- \sum_{j \in jobs} (1 - health(j)) \cdot$ QoS agreed penalty
	$- \sum_{h \in Hosts, i \in cpus(h)} (pr_i[h] \cdot pwr_i) \cdot$ power price
	$- migr(schedule_{old}, schedule) \cdot time_{migration} \cdot$ Revenue Job/Hour

At this moment, note that the problem as written as shown loses its linearity as *scheduling* and *jcpu* are being multiplied as being both variables of the same problem, and this requires to re-write some details in order to obtain again a linear problem.

Having the variable schedule as a binary values matrix, the *Capacity* constraint

can be understood as

$$Capacity\{h \in Hosts\} := \sum_{j \in Jobs} [\text{ if } (schedule[h,j] = 1)\ jcpu[j] \text{ else } 0] \leq \sum_{i \in cpus(h)} pr_i[h]$$

For this, a change of variables can be performed and rewrite the constraint as

$$Capacity\{h \in Hosts\} := \sum_{j \in Jobs} quota[h,j] \leq \sum_{i \in cpus(h)} pr_i[h]$$
$$quota[h,j] = \text{ if } (schedule[h,j] = 1)\ jcpu[j] \text{ else } 0$$

This *quota* condition formulated as a set of linear constraints in the following way

$$\begin{aligned} quota[h,j] &\geq schedule[h,j] \\ quota[h,j] &\leq schedule[h,j] \cdot BigConst_1 \\ quota[h,j] - jcpu[j] &\leq (1 - schedule[h,j]) \\ jcpu[j] - quota[h,j] &\leq (1 - schedule[h,j]) \cdot BigConst_2 \end{aligned}$$

being $BigConst_i$ constant values always big enough to surpass the sum of the job requirement ranges, just assuring that the inequalities of conditions always are valid letting the model to perform the target condition. Also remember that $jcpu[j]$ is an integer greater than zero.

The final integer linear program stays as follows:

Variables: $schedule[Hosts, Jobs]$, as Integer Binary ; representing the Schedule
$quota[Hosts, Jobs]$, as Integer ; representing CPU quota for each job in each host
$pr_i[Hosts]$, as Integer Binary ; representing the use for each host of its i-essim CPU
$jcpu[Jobs]$, as Integer ; representing the CPU usage of job j

Parameters: pwr_i, as the power consumed by an i-essim CPU
$cpus(h)$, as CPUs existing in host h
$consmin(j)$, as the minimum required CPU consumption of job j
$consmax(j)$, as the maximum required CPU consumption of job j

Functions: $migr(schedule_{old}, schedule)$, as $\frac{1}{2} \cdot (schedule_{old} \oplus schedule)$ (w.o. leaving or new jobs)
$health(j)$, level of QoS of the job j result of $jcpu[j] \sim \langle consmin(j), consmax(j) \rangle$

Maximize:
$$+ \sum_{h \in Hosts, j \in Jobs} (schedule[h,j]) \cdot \text{Revenue Job/Hour}$$
$$- \sum_{j \in jobs} (1 - health(j)) \cdot \text{QoS agreed penalty}$$
$$- \sum_{h \in Hosts, i \in cpus(h)} (pr_i[h] \cdot pwr_i) \cdot \text{power price}$$
$$- migr(schedule_{old}, schedule) \cdot time_{migration} \cdot \text{Revenue Job/Hour}$$

Constraints:
$$Procesors\{h \in Hosts, i \in cpus(h)\} := pr_i[h] \geq pr_{i+1}[h]$$
$$Unique\{j \in Jobs\} := \sum_{h \in Hosts} schedule[h,j] \leq 1$$
$$MaxCPU\{h \in hosts\} := \sum_{i \in cpus(h)} pr_i[h] \leq cpus(h)$$
$$Capacity\{h \in Hosts\} := \sum_{j \in Jobs} quota[h,j] \leq \sum_{i \in cpus(h)} pr_i[h]$$
$$MarginCPU\{j \in Jobs\} := 0 < consmin(j) \leq jcpu[j] \leq consmax(j)$$
$$QoSAux1\{h \in hosts, j \in Jobs\} := quota[h,j] \geq schedule[h,j]$$
$$QoSAux2\{h \in hosts, j \in Jobs\} := quota[h,j] \leq schedule[h,j] \cdot BigConst_1$$
$$QoSAux3\{h \in hosts, j \in Jobs\} := quota[h,j] - jcpu[j] \leq (1 - schedule[h,j])$$
$$QoSAux4\{h \in hosts, j \in Jobs\} := jcpu[j] - quota[h,j] \leq (1 - schedule[h,j]) \cdot BigConst_2$$

4 Evaluation of the Method

In this section the method is evaluated, exposing the scenario used in order to test the approach. Also all the decisions taken in order to set up a representative testbed are exposed, building a good scenario to focus on the method while letting the addition of new elements, variables, or ideas.

4.1 Programming and Simulation Environment

The experiments done in order to test this approach have been performed simulating a real workload and also simulating a data-center formed by different sized testbed machines. In this occasion, as the important thing to be evaluated and tested is the methodology and algorithms for scheduling and making decisions, a real data-center and a scheduling mechanisms have been recreated in R [9], extracting the behavior formulas and data-center working modules directly from the cloud simulator EEFSIM [6, 11] made by R.Nou, F.Julia, O.Fito and I.Goiri. The EEFSIM is a cloud simulator oriented to power and performance experimentation, with capabilities of simulating multiple data-center clouds (or single data-center), with heterogeneous physical and virtual machines, also heterogeneous resources and network connections.

For this approach, the modules and formulas have been implemented in R, using workloads and data-center configurations generated from real loads and examples. The example simulated data-center is presented in Table 4.1.

Number of Hosts	CPU	Memory
20	4 @ 3GHz	4 GB
10	2 @ 3GHz	4 GB
10	1 @ 3GHz	4 GB

The used workload corresponds to a transactional workload on application web-servers, obtained from a real web-applications workload. These web-applications simulate customers who wants to run their services on top of the the provider. The behavior of these applications deployed corresponds with the one of SPECweb2009 e-Commerce application [2] which is used as web-based application and its model has been obtained by stressing this application (deployed on a Tomcat v5.5 with an hybrid architecture) with different input loads and with different processing units. The details of the workload are shown in Table 1. This modeling, as proposed in

Type	SLA type	#VMs	Description	Mean duration
Web	Performance	10	Monday	86400”
Web	Performance	10	Tuesday	86400”
Web	Performance	10	Wednesday	86400”
Web	Performance	10	Thursday	86400”
Web	Performance	10	Friday	86400”
Web	Performance	5	Saturday	86400”
Web	Performance	5	Sunday	86400”
Web	Performance	20	One week	604800”

Table 1: Workload details

[11], is focused on the response time high-level metric and, relating this metric with both incoming users load and CPU usage of the server. The modeling details include an *stationary* response time (RT), when the incoming load causes a CPU utilization less than 60%; a *slightly increased* RT when this CPU utilization is between 60% and 80%; and a *polynomial* behaved RT when the server is overloaded.

The power consumption is also determined in EEFSIM, where the real power consumption is measured using different workloads in a 4-CPU computer whose kernel includes some energy efficiency policies. As seen in previously the wattage increases with the CPU requirements on a physical machine, but this increment is lower with each extra processor used on a machine. This is the reason why consolidation optimizes the power consumption. Also, in the experiments of this approach the turning on and off of used and idle machines is considered to reduce the consumption, as seen in works like [8, 3].

In order to set the economic values of each involved element, the EEFSIM and its related works (as seen below) established that providers behave as a cloud provider similar to Amazon EC2, where users will rent some VMs in order to run their tasks. The pricing system for the VMs is similar to the one EC2 uses and medium instances with high CPU load are assumed, which have a cost of 0.17 euro/hour (EC2 pricing in Europe). Also the electricity pricing used is the Spanish one, that is 0.09 euro/KWh [1]. The VM migration process can be performed in several ways: as stopping the service and resuming it after full VM data is copied, or creating a copy into the physical destination before deleting the original and then change the load flow. Anyway, this process may cause in some occasions a interruption of service or just degrade the user experience for a short period of time, and as example of a simple valid SLA in this model is proposed a *migration penalty* consisting in a economic compensation to the client (i.e. the execution during a migration period becomes for free). For the present approach, the maximum migration time is set to 5 minutes, while each scheduling round is set to 1 hour, according to the variability of the performed workloads.

As a parameter defining the Quality of Service at low level, the concept *health* is defined as the ratio between obtained CPU and the required CPU in function of the client load. The maximum loss of QoS permitted can be determined in each job SLA, but as an example and case of study here a range $[0.8, 1.0]$ is determined as acceptable offered CPU, and a penalization of $(1 - \frac{CPU_{job}}{OfferedCPU_{job}}) * 0.17 euro/hour$ (price of job per hour) is applied in order to add a benefit factor. Obviously these parameters can vary in function of each job SLA adjusting the minimum accepted health factor, the way of calculating the health factor, and the price of the job.

Finally, in order to find good scheduling results implementing the Integer Linear Programs, different popular implementations of the simplex algorithm [4] have been used. For the exposed results shown in next subsections the GNU Linear Programming Kit (glpk) has been used, but other options have been run in order to find enhanced simplex algorithms like lpsolver or CPLEX [10]. Here we will not discuss a comparison between them, as lpsolver often find better solutions than glpk with the same running time and CPLEX is able to find optimal solutions in seconds instead of minutes like the other ones. The glpk has been selected just because is the easiest to apply and experiment with, without a license requirement (i.e. CPLEX requires a limited payment license to be run).

4.2 Experimentation

In this subsection the method is tested on each of its factors (power, economic, qos). The integer linear program can be compared in an *only-power* version, looking for minimize the power consumption; in an *economic versus power* version, looking for increase the revenue minus the power economic cost; and in the full version, including the tolerance to some loss of QoS in favor to the power saving.

As expected, the experimentations reveal that the power model is the one that performs the maximum number of migrations as its priority is to reduce the consumption and in a natural way the scheduler consolidates the jobs at each round. This model only contemplates options that reduce the number of processors used and machine on-line, and this brings the policy of migrating jobs as many times as required. Without any restriction referring to migration penalization, this model is the one that performs best in an energetic way. Economic and QoS loss tolerated models do not optimize as good in energy, mainly because they include restrictions at applying this migration policy. The power consumption is shown in Figure 1.

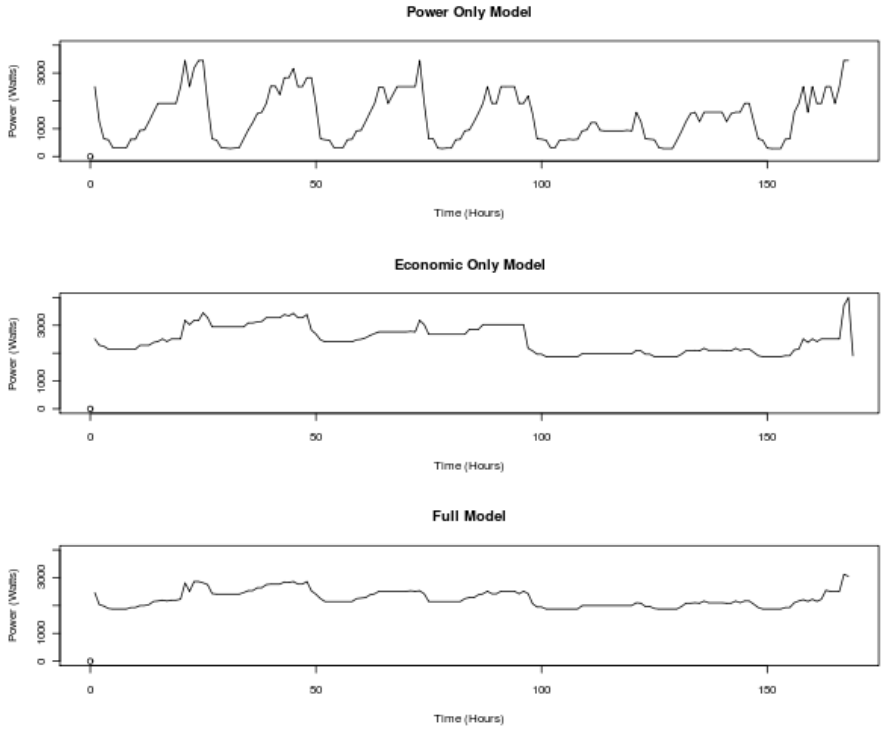


Figure 1: Power consumption in for each model

Economic only and QoS loss tolerated models include the policy of reducing the number of migrations by penalizing them. This makes that their power saving is reduced compared with the power only model, but here the schedule is reducing the number of times a job can be temporally "out of service". As Figure 1 shows, economic model and QoS loss tolerance model power consumption is similar, also QoS tends to improve the economic only model, as lets the reduction of resource quotas to improve power saving.

When introducing the explained migration penalty the optimization changes as the economic based models improve the revenue brought by maintaining machines running without being migrated. As seen in Figure 2, power model degrades in revenue as migrations are done without restrictions. The model tolerating some QoS loss maintains similar revenues than the economic model reducing the number of migrations by reducing the quota of resources, also penalized by the explained *health rule*. This temporary reduction of resources in order to reduce the migration needs is shown in Figure 3, where clearly QoS loss tolerated model nearly avoids migration.

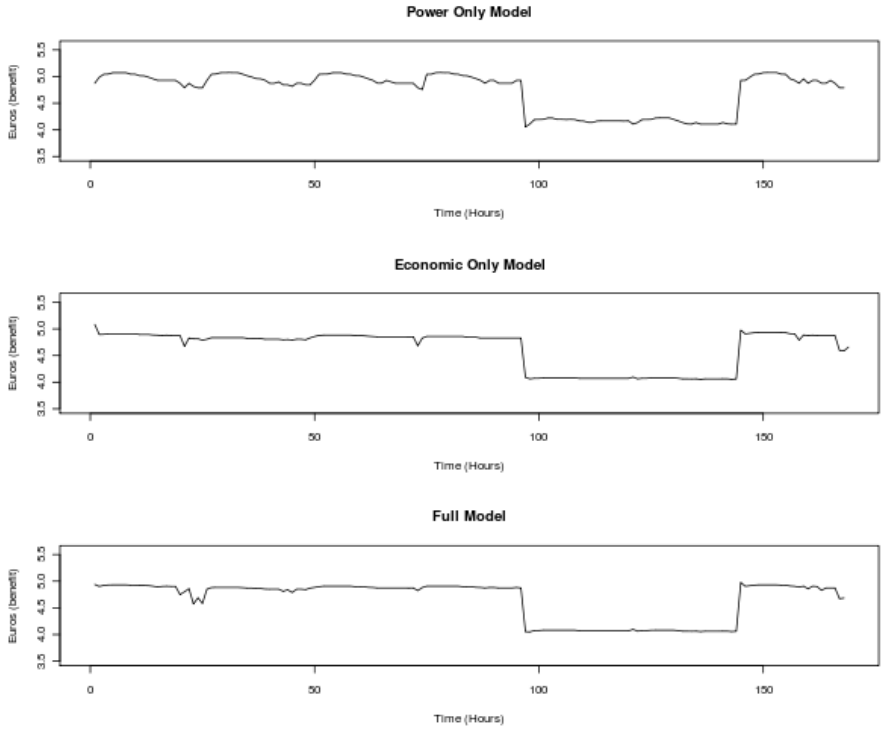


Figure 2: Benefit obtained from each model

It is important to remark the fact that the reason of penalizing the migrations is that each migration brings a chance of interrupting the service during a brief period of time or any other drawback of this technology like having duplicated services or jobs. For this reason economic and QoS models should be preferred than power-only models, as they attempt to reduce the number of chances of service interruption. For future models, whether knowing the exact migration specifications, this chances and exact SLA penalization could be added to the optimization model, so the decision of migrating a job versus do not do it would be better weighted assuming a more accurate probability of interruption of service.

As said before, applying a reduction of resources in order to avoid migrating supposes a loss of quality of service. As shown in Figure 4 the loss of QoS while compressing jobs in machines usually do not imply a great loss of QoS, translated in response time for the current case of study workload. The average health found is 0.998, and in very few cases the health of the jobs in a physical machine reaches

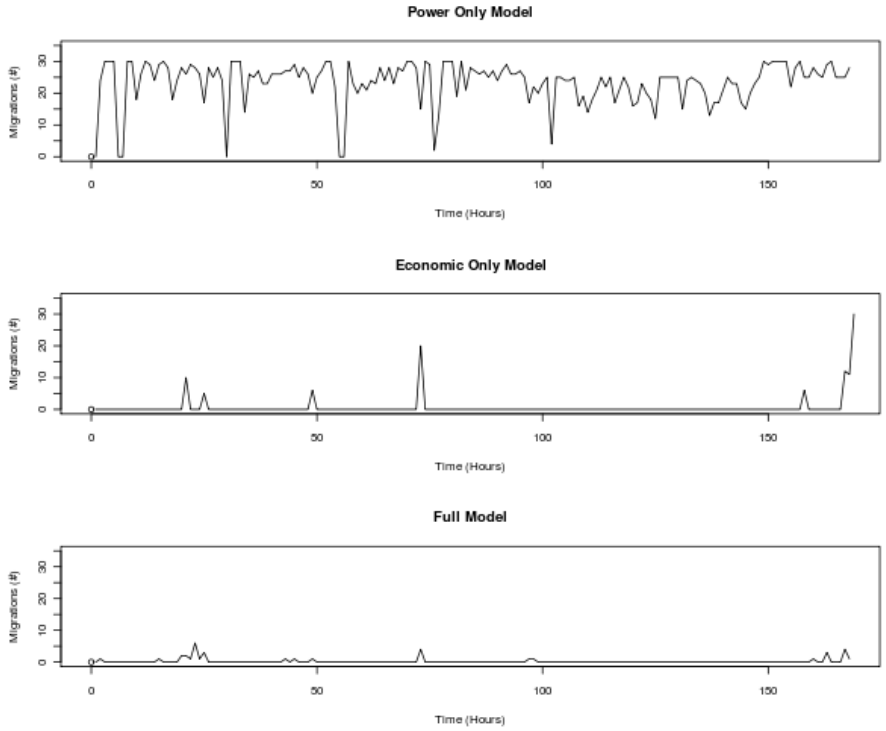


Figure 3: Number of migrations for each model

the minimum threshold.

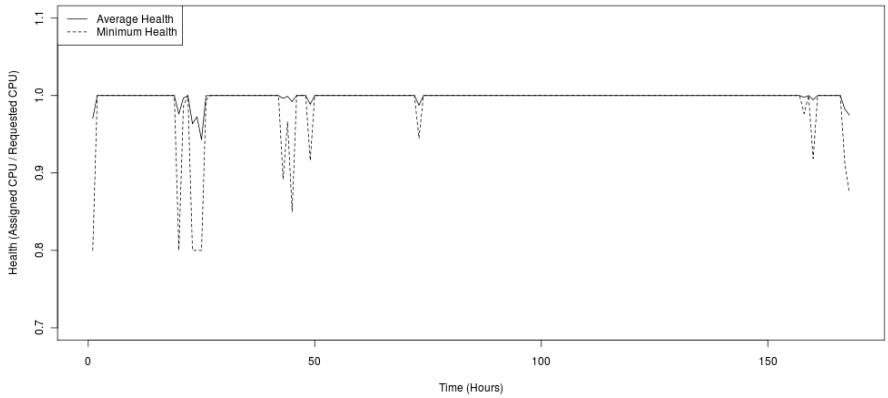


Figure 4: Levels of Health in tolerated QoS loss model

Table 2 summarizes the consumption, benefit and number of migrations performed by each model. In the table a Round Robbin scheduling algorithm has been added in order to compare with a naive scheduler. This Round Robbin algorithm performs a "random" allocation always assuring that the job fits into the target machine.

The coefficients applied in the problem weighting each task execution, the power price, the migration penalty and the QoS penalty define the policy to be followed in

Model	Power cons. (watts)	Benefit (euro)	Migrations (#)
Round Robbin	1687717.0	598.1375	4712
Power Only	236398.3	794.5513	3944
Economic Oriented	419813.1	782.7054	100
QoS relaxation	376801.8	780.8093	35

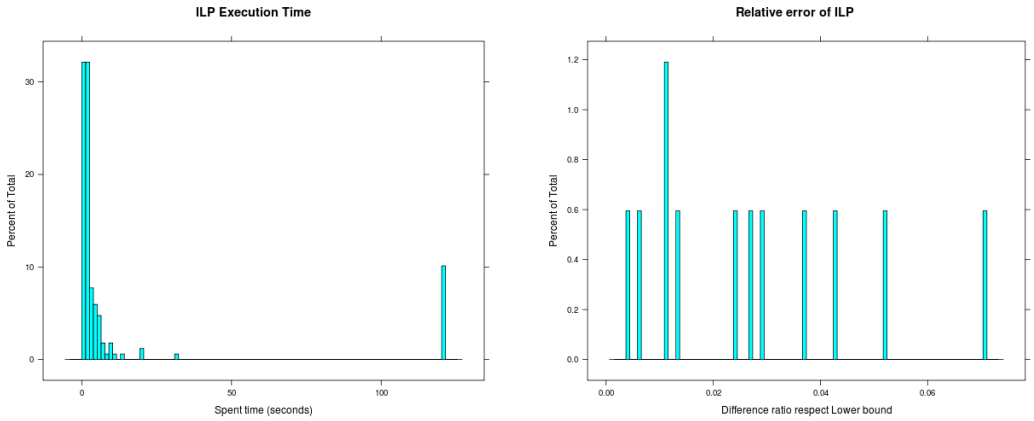
Table 2: Comparative between models

order to optimize. This case of study assumed a policy that obviously can change when prices change, and when technology changes. If migration technology is improved reducing the maximum migration time, the penalty of migration can be reduced so in some occasions moving jobs will be better than sacrificing jobs performance. Furthermore, a modeling of each kind of job could provide a characterization of the probability of "out of service" when migrating a specific job and adjust the migration penalty depending on the job to be moved. This also would provide an accurate selection of which jobs are more candidates to be moved when the situation requires it. The same occurs with the QoS threshold, being modified for each job and circumstance, so some jobs are more or less able to be economized depending on their execution time. Here a *job migration vs job compression* situation is faced, depending entirely of the policy applied at each moment.

Integer Linear Programming can be used directly as a scheduling algorithm in some occasions, when engineers and designers know that the situations to control will be solvable in short time (seconds or minutes). But unfortunately, simplex algorithm can become exponential, and the integer optimization usually is exhaustive, and in some situations finding a good enough solution requires excessive search time. For this case of study the average time in order to find a good solution: an average of 0.2447% difference respect the simplex lower bound [min 0.0%, mean 0.2447%, max 7.0530%], with a mean of 14 seconds time search for scheduling often 30 jobs x 40 machines (with a maximum search time of 2 minutes, not often reached), as seen in Figures 5(a) and 5(b). So, this ILP can be used to find and discover new heuristics, comparing them to the best (or sub-optimal) solutions found with ILP.

4.3 Impact of Policies

Finally by looking at the proposed model, each element included in the benefit function can be seen as an element of a policy system. Having Benefit B, Revenue R, Migrations M, Loss of QoS Q and Power P as: $B = R - M - Q - P$, the elements M and Q represent the policy of the system in front of the possibility of migrations and task resources compression. The impact of each policy can be measured by testing them using multipliers in order to check how much vary the power and real benefit while varying the enforcement of the migration penalty or the loss of QoS penalty. Setting $\tilde{B} = R - (\lambda M + \varepsilon Q) - P$ and a policy $\Pi = \langle \lambda, \varepsilon \rangle$ indicating how each element will be weighted in the ILP solver, some tests are run to obtain the relation between each migration and loss of QoS for each power saving unit. A first set of executions test the migration policy by varying its weight (λ) and forbidding the loss of QoS ($\varepsilon = \infty$), while other set of executions test the tolerance to QoS

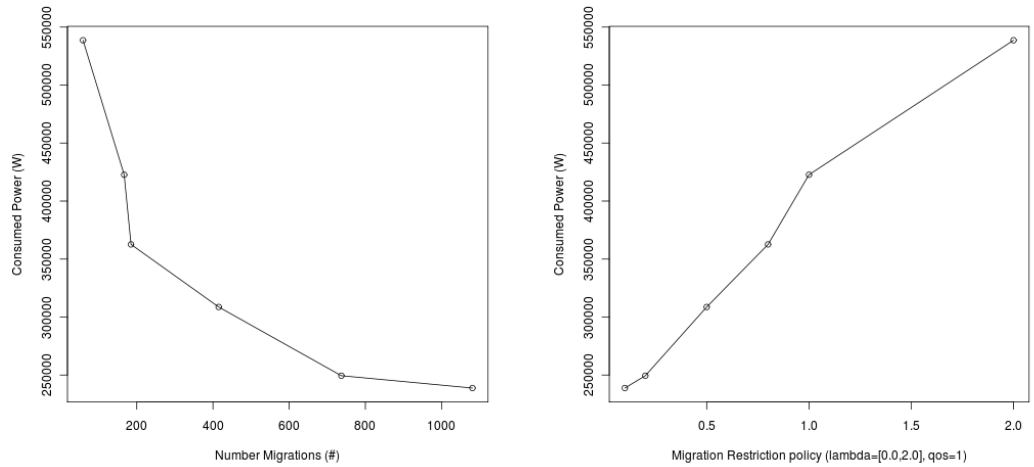


(a) (b)

Figure 5: Measures of performance of the ILP solver

loss by forbidding the migrations ($\lambda = \infty$) and varying the weights of the QoS loss penalty (ε) and letting no minimum bound to the QoS loss.

In the case of the migration policy test, it is obvious that the migration technique is always applied in order to reduce the number of running processors in the whole system, so as shown in Figure 6(a) the number of migrations is reversely related to the consumed power. This can be understood as "migrations relief power consumption, while paying a price", and it is the relation between the cost per kWh and the penalty for each migration. If the price of the amount of power a migration can save is bigger than the cost of performing this migration, do it is better. So policies based in migration can modify the migration penalty in order to adjust the number of migrations to be performed in front of a determined amount of power to be saved, as seen in Figure 6(b).



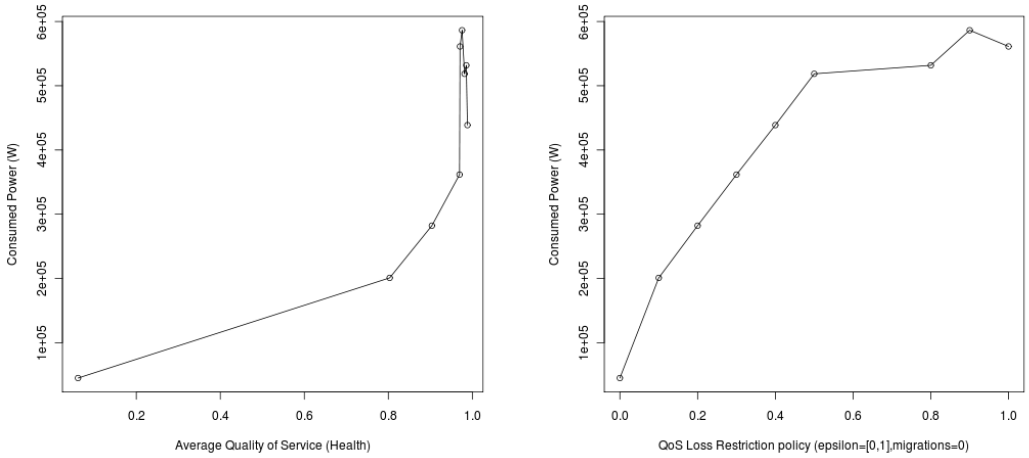
(a) Power vs Number of Migrations (b) Power vs Migration Restriction

Figure 6: Power versus migrations and migration restriction policy

Finding the best fitting linear regression from the power vs. migration results,

the generated model matches with the form: $\hat{Power} \simeq MinimumRequiredPower + (maxPossibleMigrations - \#Migrations) * MachineBasePower$. This means that in average for each migration performed, the power of having on-line one machine is saved, demonstrating that the method performs first that migrations that improve most the power saving. In this case, performing a migration that allows to shut down a machine is preferred to perform a migration that just saves a few watts. From Figure 6(a) we can distinguish three parts, the first one where the policy optimizes only with migrations that directly shut-down machines (0-200 migrations), the second one where migrations shut-down machines and reduce the number of running processors (200-800 migrations), and the third part where saturation comes and no for more migrations done the power will be easily reduced due to reaching the minimum required power.

In the case of Power versus Loss of QoS it can be observed that the power is related to the level of Quality of Service (in terms of resource occupation) and the restrictions to be applied on QoS loss, as seen in Figures 7(a) and 7(b).



(a) Power vs QoS (health) level

(b) Power vs QoS Loss Restriction

Figure 7: Power versus QoS (health) and QoS Loss restriction policy

Note that the policy values λ and ε are directly related to the number of migrations and QoS respectively. So in tunning the model establishing the weight of each condition (or policy or price), a trade-off between the provider revenue per application and the penalizations for each migration and loss of QoS. The ILP model presented here is complemented by finding these penalization weights by applying the policy that minimizes the power consumption while granting a good QoS level and reducing the possibilities of service outage caused by migrations. Also with this information the model can be adjusted if business policies decide not to allow more than X migrations per time unit, or not to set a determined average of QoS allowed (or also a minimum QoS level without restricting it in the ILP constraints).

5 Conclusions

In order to optimize the scheduling and management of data-centers several factors to keep in mind, like the economic ones (obtaining revenues from task executions and resource usage), bringing good service quality to the customers and clients, and also the energy-related factors in order to run the infrastructures. This optimization problem can be modeled as an integer linear program, so ILP methods can help in order to solve it in a relative good time, or also help to find accurate models in order to test other methods.

Taking the advantages of virtualization and consolidation techniques, this model can address policies that focuses on energy-saving goals, like using migration to turn-off unnecessary machines, or lend tasks fewer resources in order to save power. The results obtained demonstrate that the ILP model allows to find optimal (or near-optimal) solutions in fast times bringing good QoS levels (by giving tasks the required resources), and each new technique introduced in form of rules and constraints allows the scheduler new options to improve the results.

Also, from the obtained results a common sense fact can be observed: in order to maximize economic benefit the usually most "sacrificed" factor is the cheapest one. In this case it is power consumption. When the schedules attempts to make a decision that maximizes revenue while not degrading QoS, migrating as minimum as possible and consuming low power, as power consumption is economically weaker than the revenue obtained by the rest of the system, it receives less attention and so benefit increasings response to higher power consumptions.

A way to adjust this power spending is to play with policies, adjusting weight to power-saving techniques in order to relax their usage penalties, so here the system would risk some economical benefit or QoS in order to save some energy. Also policies decide which kind of risks are to be taken between chances of service outage caused by migrations or chances of QoS loss by reducing the given resources to tasks. Future work is addressed to research in the way of decision makers and modeling the kind of risks to be taken at each moment, so static policies can turn into dynamic depending on the context, the load and the interests of providers and clients.

References

[1] Europe's energy portal. http://www.energy.eu.

[2] SPECweb2009 E-commerce workload, 2009. http://www.spec.org/web2009/.

[3] J. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavalda, and J. Torres. Towards energy-aware scheduling in data centers using machine learning. In *1st International Conference on Energy-Efficient Computing and Networking (eEnergy'10), University of Passau, Germany, April 13-15*, pages 215–224, 2010.

[4] G. B. Dantzig and M. N. Thapa. *Linear programming 1: introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[5] J. O. Fitó, Í. Goiri, and J. Guitart. SLA-driven Elastic Cloud Hosting Provider. In *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and*

Network-based Processing (PDP'10), Pisa, Italy, February 17-19, pages 111–118, 2010.

[6] Í. Goiri, J. Guitart, and J. Torres. Characterizing Cloud Federation for Enhancing Providers Profit. In *Proceedings of the 3rd International conference on Cloud Computing (CLOUD 2010), Miami, Florida, USA, July 5-10*, pages 123–130, 2010.

[7] Í. Goiri, F. Julià, J. Ejarque, M. De Palol, R. M. Badia, J. Guitart, and J. Torres. Introducing Virtual Execution Environments for Application Lifecycle Management and SLA-Driven Resource Distribution within Service Providers. In *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (NCA'09), Cambridge, Massachusetts, USA, July 9-11*, pages 211–218, 2009.

[8] Í. Goiri, F. Julià, R. Nou, J. Berral, J. Guitart, and J. Torres. Energy-aware Scheduling in Virtualized Datacenters. In *Proceedings of the 12th IEEE International Conference on Cluster Computing (Cluster 2010), Heraklion, Crete, Greece, September 20-24*, 2010.

[9] K. Hornik. The R FAQ, 2010. ISBN 3-900051-08-9.

[10] IBM. Solver cplex, 2003. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/ (accessed 17 September 2010).

[11] F. Julià, J. Roldàn, R. Nou, O. Fitó, Vaquè, G. Í., and J. Berral. EEFSim: Energy Efficency Simulator, 2010.

[12] I. Kamitsos, L. Andrew, H. Kim, and M. Chiang. Optimal Sleep Patterns for Serving Delay-Tolerant Jobs. In *1st International Conference on Energy-Efficient Computing and Networking (eEnergy'10), University of Passau, Germany, April 13-15*, 2010.

[13] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, and Y. Chen. GreenCloud: a New Architecture for Green Data Center. In *6th International Conference on Autonomic Computing and Communications, Industry Session, Barcelona, Spain, June 15-19*, pages 29–38. ACM, 2009.

[14] R. Nathuji, K. Schwan, A. Somani, and Y. Joshi. Vpm tokens: virtual machine-aware power budgeting in datacenters. *Cluster Computing*, 12(2):189–203, 2009.

[15] V. Petrucci, O. Loques, and D. Mossé. A Dynamic Configuration Model for Power-efficient Virtualized Server Clusters. In *11th Brazillian Workshop on Real-Time and Embedded Systems (WTR), Recife, Brazil, May 25*, 2009.

[16] Y. Tan, W. Liu, and Q. Qiu. Adaptive power management using reinforcement learning. In *ICCAD '09: Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 461–467, New York, NY, USA, 2009. ACM.

[17] W. Vogels. Beyond server consolidation. *Queue*, 6(1):20–26, 2008.