CHAPTER 8

TOWARD ENERGY-AWARE SCHEDULING USING MACHINE LEARNING

JOSEP LL. BERRAL, IÑIGO GOIRI, RAMON NOU, FERRAN JULIÀ, JOSEP O. FITÓ, JORDI GUITART, RICARD GAVALDÁ, and JORDI TORRES

8.1 INTRODUCTION

The cloud and the Web 2.0 have contributed to *democratize* the Internet, allowing everybody to share information, services, and IT resources around the network. With the arrival of digital social networks and the introduction of new IT infrastructures in the business world, the Internet population has grown enough to make the need for computing resources an important matter to be handled. While few years ago enterprises had all their IT infrastructures in privately owned data centers, nowadays the big IT corporations have started a *data-center race*, offering computing and storage resources at low prices, looking for outside companies to trust them for their data or IT needs.

A single web application in the cloud can be easily used by people from around the world, so data and computation need to be available from everywhere, having in mind things such as the quality of service (QoS) and the service-level agreements (SLAs) between users and servers. Services offered by Google and YouTube, for example, 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. Given the amount of applications running now on the cloud and the amount that will come, coordinating all its applications, resources, and services becomes by itself a hard optimization problem.

Energy-Efficient Distributed Computing Systems, First Edition.
Edited by Albert Y. Zomaya and Young Choon Lee.

8.1.1 Energetic Impact of the Cloud

Having powerful enough data-centers to server applications or computation time is not the only thing to keep in mind when building the Cloud. As energy-related costs have become a major cost factor for IT infrastructures and data centers, power consumption has become an important element to keep in mind when designing and managing them. This energetic cost is reflected in the electric consumption, which is sometimes nonlinear with the capacity of that data centers. It also has direct environmental impact and is conditioned by social pressure for efficiency. Companies dedicated to cloud-based services, and the research community are being challenged to find better and more efficient power-aware resource management strategies.

Until now technological improvement sufficed to cover the increasing IT demand, bringing faster processors, bigger storage devices, and faster connections between resources. The energetic factor was not relevant enough to be focused on. Now the demand is growing faster than technological improvement, so each time we need bigger data centers to be cooled down in colder places, having enough power supply [1].

Reaching an optimal performance of cloud services and resource management requires an *intelligent management*, conscious of the importance of each resource used, each service given, the way power is consumed, and the relation between power consumed and work done. This intelligent management complements the technological improvement, allowing better resource use, borrowing and lending resources when it is convenient to do so, and improving the QoS without scaling up the data centers unnecessarily.

8.1.2 An Intelligent Way to Manage Data Centers

This intelligent management would be easy if the manager knew in detail the structure and elements in the cloud, if system administrators could keep constantly watching the system, and if experts could advise what to do in each situation. Unfortunately this is often not possible. Unfortunately, this is not often possible. The cloud, as its name suggests, becomes an abstract cloud of resources. Each domain of resources has its own resource broker and interface for dealing with resource borrowers and lenders, so a part of the cloud cannot manage or get all the information from other parts of it.

Also, systems running on the cloud are hard to model, as well as predict. There are no experts in some applications of the cloud, and some predictive variables are hidden to the naked eye, so it is very difficult to predict the behavior of the whole (or just a part of) system when lots of variables are involved. Furthermore, keeping a human operator watching over all events and resources of the system; reacting to each change when changes happen so fast; and executing the best solution each time is not possible. Intelligent management must be automated, must “understand” what is happening in the system, and must “learn” about actions to be taken.

8.1.3 Current Autonomic Computing Techniques

Current data centers and large-scale distributed computing systems are increasingly implementing the techniques of autonomic computing. Automation on large-scale systems has become a hot spot on system improvements, letting the systems manage themselves (self-healing, self-protection, self-optimization, and self-configuring) from expert systems, statistic models, and ad hoc rules.

The intermediary software (also known as *middleware*) in charge of performing these autonomic computing techniques requires models that capture the most important factors of the systems while allowing abstract reasoning. The models must allow formalizing behaviors and interactions that help the use of optimization techniques (from simple heuristics to complex techniques) based on, that is, what-if predicting techniques or expectation formulas for action results. It is important to remark that optimizations at different system levels interfere between them. This makes the behavior of the current systems unmanageable at execution time, requiring novel optimization techniques that implement self-properties at runtime. These autonomic techniques must be developed to manage workload fluctuations and to determine optimal trade-offs between performance and energy costs.

All these solutions can also be improved if the system learns from itself, becoming itself an expert, modeling from statistics, and writing and improving its rules or management policies. ML (and the closely related field, data mining) brings a set of methods and ideas to, given a set of observations from the system, infer and induce the behavior of the system. Also, these methods are often easy to update in front of changes, or just or general enough to accept changes. This ability to learn for improving the performance of large-scale systems opens a new wide research area combining the self-capabilities of autonomic computing and the capabilities of discovering knowledge from systems.

8.1.4 Power-Aware Autonomic Computing

Middleware requires new advanced management mechanisms to provide the necessary control actuators to successfully manage the resources in order to add energy efficiency as an operating parameter. Nowadays, the most common techniques used in the research literature of the area can be summarized as virtualization, turning on/off servers, DVFS (Dynamic Voltage and Frequency Scaling), and hybrid nodes/hybrid data centers.

- Virtualization is key to reducing power consumption. With virtualization, multiple virtual servers can be hosted on a smaller number of more powerful physical servers, using less electricity. Virtualization mechanisms are currently used for consolidation.
- Turning on/off servers reduces the overall consumption through consolidation. As reduction of needed resources is the goal of consolidation, shutting down of these resources when possible is where actual energy saving is achieved.

- DVFS is the reduction of voltage and frequency, providing substantial saving in power at the cost of slower program execution. Current microprocessors allow power management by DVFS.
- In hybrid data centers, it is possible to choose among a variety of resources, depending on the system load and requirements taking into account energy consumption. Even more, we would like to have the system (not a human supervisor) choose and learn to choose among different resources with the same final functionality but with different characteristics.

Usually, all these techniques can also be combined, improving the level of consolidation and effectiveness. By deciding to turn the physical machines on and off when virtualized machines are consolidated or specializing the physical machines where virtual machines (VMs) are going to be consolidated, power saving can be improved.

8.1.5 State of the Art and Case Study

In this chapter, a brief survey of the state of the art of "intelligent management" and power-aware techniques is shown in Sections 8.2 and 8.3, focusing on the works that are introducing machine learning and other artificial intelligence techniques. Also, a case study summarizing our experiences, applying some of these techniques is introduced in Section 8.4, explaining some practical applications of each technique and showing results and conclusions on the application of the learning mechanisms on a self-management system.

8.2 INTELLIGENT SELF-MANAGEMENT

Adaptive and updatable mechanisms have been developed in order to optimize the management of the cloud and improve the resource usage and the QoS. But the cloud is becoming more complex and application requirements are increasing and knowledge-based and data mining techniques are starting to be applied. Once information about the execution, resources, and requirements is available, artificial intelligence (AI) and machine learning (ML) can be applied to improve prediction and information retrieval, letting the system make some decisions with more autonomy and with more accuracy.

In *intelligent management*, there are different techniques that are beginning to be researched and applied. The first ones are the standard AI-based techniques. These techniques use prediction and heuristic algorithms in order to anticipate system performance and act in consequence. Fuzzy logic, genetic algorithms, and other AI methods are used in order to improve QoS, resource allocation, and execution of applications. The second ones are the ML-based applications. These techniques use the recorded information from past behaviors to create a model that best fits the usual behavior and lets the system detect anomalies and make decisions over the system.

8.2.1 Classical AI Approaches

AI methods have been historically defined as *a machine thinking like human* methods, but nowadays, AI is more like *finding a suitable/intelligent solution with limited time/space*. The classical AI methods are about searching a good solution to a problem on a representation space, or representing knowledge using ontologies and expert systems, and all of these as optimal as possible. Exhaustive searches on a representation space are often NP-hard or exponential problems, and the same happens with many ontology systems or huge expert systems, so searching methods using heuristics, genetic algorithms, and fuzzy knowledge techniques are used to perform these searches in a viable way, not finding always the best solution but having a suboptimal solution in the available space and time. The solution is intelligent in the sense that it does not examine the whole space of solutions.

8.2.1.1 Heuristic algorithms. With the capability of using knowledge and heuristic algorithms, it is possible to predict some situations with good accuracy. This prediction can be applied to detect unwanted situations or behaviors or to view the near future situations such as imminent changes in the workload, changes in the resource demand, or limit situations of resource offers. In approaches such as those presented by Vraalsen [2] and Fahringer [3], some prediction models for parallel programs and grid-based applications are presented, where a method based on heuristics for predicting application performance is presented. With these methods, the system looks for detecting unexpected behaviors, usually caused by unanticipated load on shared grid resources. Once the heuristics detect these unexpected execution behaviors, a fuzzy logic-based algorithm is used to check and decide how to maintain the QoS of each execution. The fuzzy logic algorithm uses the information monitored from the application execution sensors and the performance contracts for the application based on an application signature model to decide what action must be taken.

8.2.1.2 AI planning. Furthermore, we can find AI techniques not only predicting behaviors or situations but also managing workflows among machines of a distributed system. These methods are basically AI planning, methods for planning, and scheduling events using as guide a set of operators and a set of observations. Some works done by Deelman et al. [4] and Gil et al. [5] show methods for scheduling jobs on a grid environment generating, for each job, resource requirements and available resource workflows. These workflows are searched and used by AI planning methods to schedule the application execution matching resource requirements with resource availability.

8.2.1.3 Semantic techniques. Another AI approach to improve the management and adaption of applications is to use semantics. On the basis of the principles of the semantic representation systems, there are some ideas presented about ordering grids and clouds toward an architecture in which information and services are given a well-defined meaning, thus better allowing computers and

people to work in cooperation. Acting above syntactic or static valuable rules, by relating systems, resources, and applications, some approaches look for ordering the cloud using logical and coherent matchings between user, resource, and application. These approaches are no more than ideas and research challenges yet, and there are some skeptics about the viability of semantic processes in high performance computing (HPC). Although some works and outlines can be found in the overview done by De Roure et al. [6] and Ejarque et al. [7], exposing the current research on the ontology-based resource management is shown in this chapter.

8.2.1.4 Expert systems and genetic algorithms. There are a lot of approaches dedicated to use expert systems and genetic algorithms for self-configuring systems. All these approaches are not directly oriented to the management of clouds or distributed systems, but they give the idea of AI methods for automatically configuring and optimizing a tasks system, such as the ones by Wolpert et al. [8], Sirlantzis et al. [9], and Rahman et al. [10]. These methods, combining classifiers with AI methods, can be used as reference for the management and planning of autonomic systems and clouds.

But AI approaches usually have to be built and validated by experts, or once a model is found, it is very difficult to renew it, and the search process must be repeated. Machine learning techniques supply a new vision of the modeling process, letting us to find these system models without the explicit requirement of experts and the ability of being able to update the model in a more easy way.

8.2.2 Machine Learning Approaches

Although some AI planning and other AI methods have been created to self-optimize or self-configure grid and cloud executions, some problems are related to lack of adaptation, uncertainty of the model, or extreme complexity of the system to build a model by experts or single patterns. To solve these problems, one very relevant solution is machine learning, which allows the creation of decision and classification models of very complex systems from the examples of the same system, in an easy way to update that models, or being possible to make a model that is general or specific enough, extracting the knowledge directly from the system execution and its environment.

Machine learning mines for data, obtaining the relevant information and attributes from it, creating the model that explains the system, and finally using the model to make decisions. The machine learning techniques consist, generally, on collecting a *training data set* from the system with data composed of system values and response attributes and creating a model through induction, able to explain these examples, expecting that new data will fit in.

The machine learning techniques are divided into supervised learning (such as classification and regression), unsupervised learning (discover the relationship between the input data), clustering (find similarities on data), and reinforcement learning (select the best decision from the past experiences and feedback). Here,

we discuss briefly about instance-based learning techniques as part of the supervised learning and reinforcement learning (RL) as the two most applied group of techniques in grid and cloud management. Also, feature and example selection are techniques important in the learning process and also important to understand how the examined system works.

8.2.2.1 Instance-based learning. Instance-based machine learning, as part of the supervised learning techniques, allows to predict from a system and its set of resources and elements information that is often not clear at simple sight, often fuzzy, and also inaccurate, uncertain, or incomplete. After obtaining this information, we can use it to create a classifier model or regression model, and obtain from that information the one required to improve the decision making process. Then, the model is able to predict or estimate this useful information, also showing what kind of relation exists between observed data and system behavior, letting us to understand the system better.

For this prediction process, we need to choose suitable prediction algorithms, computationally light but able to obtain good results once trained with data from various workloads. Also, we need to obtain a good training set (a set of data containing labeled instances from representative executions) and another test (or validation) set. If, after training, the predictor guesses are close to the correct values on the test set, we expect that they will also be correct on future data sets. Figure 8.1 shows the basic schema of a supervised machine learning process.

Before the cloud, when all the research was based on the grid model, some methods and solutions were created for grid self-management and adaption of the system to the applications running on it. For example, some works like [11] presented a comparison of different machine learning algorithms such nearest-neighbor, weighted-average, and locally-weighted polynomial regression, upon the resource managing PUNCH framework, to model and predict application performances in order to be able to allocate or schedule the application in a grid

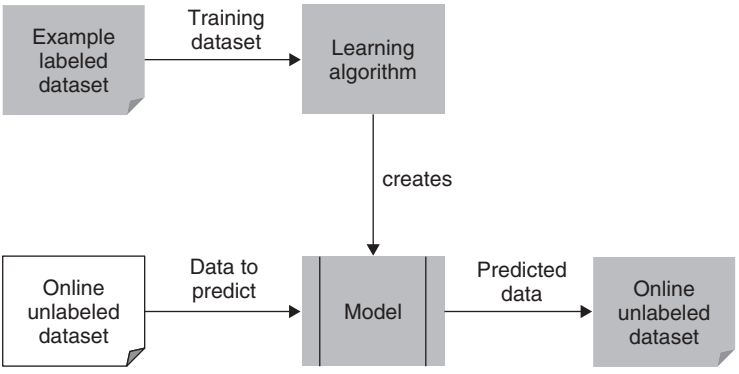

Figure 8.1 Supervised machine learning schema.

environment. Job and resource scheduling is a problem when expert systems and policy-based algorithms become expensive or too complex, so we need another cheaper and more simple system for doing it.

Other works use machine learning on self-management, focusing specifically on self-healing and fault diagnosis. Detecting failures in resources and applications, and also the root of failures, has become a very interesting area where inductive learning methods are also applied. Approaches such the one presented by Hofer and Fahringer [12] shows an application-specific fault diagnosis based on indicators, symptoms, and rules. For this approach, two techniques have been used: a supervised classification to find the reason of the failure and clustering techniques to find what failures are the result of the same cause. Other works such as those presented by Zhang et al. [13] focus explicitly on regression functions to find memory leaks. Also, works by Alonso et al. [14–16] presented a framework for monitoring a complex web application server and estimate, through learning and regression techniques, the time until fault of the server caused by resource leaking, such as memory or CPU. It also proposes a technique for detecting the root cause component of the fault. All these supervised learning techniques are usually combined with macropolicies and utility functions, where, depending on the results, a set of specific decisions are taken in order to adjust the system according to the prediction. Example given, Poggi et al. [17–19] presented a framework where, depending on user modeling predictions, machines are set up or shut down, saving energy by closing as many web servers as possible, keeping the users predicted as "customers" in the on-line machines.

Thanks to the appearance of the Weka Toolkit [20], several autonomic computing researchers have been able to introduce machine learning into their work as well as improve them. Wildstrom et al. [21–23] presented an approach for online hardware reconfiguration using algorithms for rules and decision making. Currently, researchers who wish to introduce some ML techniques into their approaches have the possibility of using that toolkit. With a better research on machine learning applications, the autonomic computing approaches will be improved in a better way.

8.2.2.2 Reinforcement learning. Reinforcement learning is the problem faced by an agent that must learn behavior through trial-and-error interactions in a dynamic environment [24]. As Kaelbling states, there are two main strategies for solving reinforcement learning problems: first, to search the space of behaviors in order to find one that performs well in the environment, by work in genetic algorithms and genetic programming and second, to use statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of the world. While supervised learning involves learning from labeled examples provided by an external supervisor, in reinforcement learning, an agent must be able to learn from its own experience.

Current self-management approaches tend to apply the second kind of reinforcement techniques, as it is easier to be applied to handle system drifts and changes. This kind of reinforcement basically consists on defining a function,

representing the system goal to be maximized. This goal usually is the benefit obtained by the system expressing all the revenues and costs of it; the resources or power consumption to be reduced; or any random variable representing weighted factors from the system expressing the interests of the system manager.

The learning process consists on learning what policies or actions must be applied given the system status, observing the results of applying them, and modifying the decision maker depending on the observed results. Policies and actions are basically operations or sets of operations done to elements from the system. These policies and actions are ranked for each situation or status by their maximum expected return for the goal function in a determined number of steps, so the decision maker selects the best ranked action given a specific status, and depending on the result, the ranking is modified. At long term, the ranks may converge to an optimal ⟨status,action⟩, whether the system does not change dramatically its configuration (Fig 8.2).

The implementation of learning algorithms is based on dynamic programming, showing the ranking function as a recursive formula, looking for the maximum return of a function at infinite steps forward. So the evaluation of each ⟨status, action⟩ pair could be defined as

$$Q(s, a) = \Sigma_{s'} E[R|s', \pi] P(s'|s, a),$$

where s is the status, π is the policy being followed, a is the action being evaluated, R is the return for the goal function, and s' is the each possible status

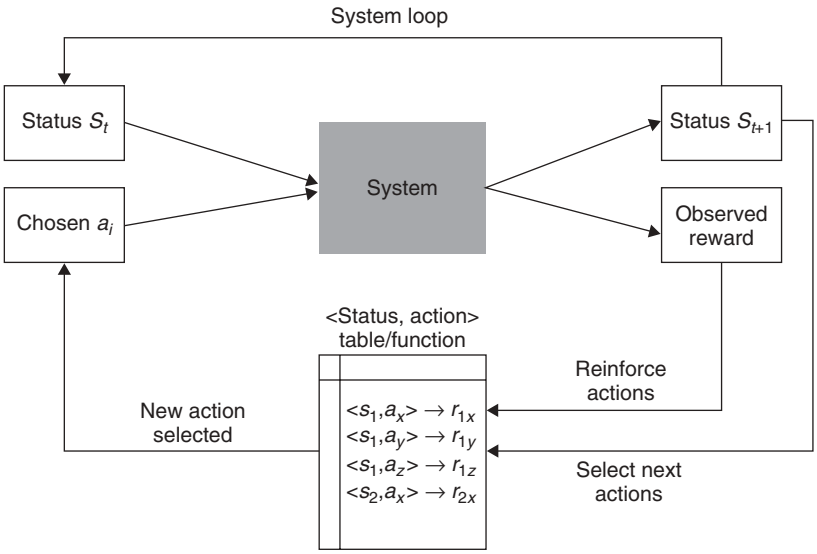

Figure 8.2 Reinforcement-learning schema.

resulting after applying a to s. The probabilities and expectations are trained by running examples and modified online with each resolution after applying the selected actions for each state. Specifically, the expectation is trained as a recursive Bellman equation

$$E[R|s_t] = r_t + \gamma E[R|s_{t+1}],$$

where the direct reward r_t and the future expected reward are weighted depending on the importance of the immediate results.

In the simplest case, the set of states and actions are discrete, and we can have a map with each estimation for each state. But as seen in the previous definition, the ⟨status, action⟩ space can be incredibly large depending on the system, as large as the amount of states and actions that can be reached and performed. This makes the problem of action selection expensive in space and in time. Thus, it is often interesting to learn the $Q(s, a)$ function using induction learning techniques, complementing the reinforcement. The expectations can be replaced by estimations, so $\hat{Q}(s, a)$ function can be trained, acting as a reward function for each action.

As a simple practical example, let us imagine a system where several actions can be chosen to be applied given a status, and from scratch, all actions are equally scored in order to obtain a good result for this status. By choosing the first random action, the system can check after if the result has been good or bad, so the score of this action will be modified by raising or degrading it. By repeating this for some iterations, the scores for each action/status will indicate what actions have been better rewarded or penalized for each status. At this point, the system can continue by choosing the most scored actions given a specific status, as well as continue evaluating them. At long term, the scorings in a stable system may converge to a stable ranking action/status. More about the mathematics and basics on reinforcement learning can be found in the works and tutorials of Kaelbling et al. [24], Sutton and Barto [25], and Bertsekas and Tsitsiklis [26].

The reinforcement-based algorithms have become recently trendy because of the potential and promising results on autonomic computing self-management, as explained by Tesauro et al. [27]. In reinforcement learning based algorithms, rules and policies can be prioritized and applied depending on the success in previous executions. As the autonomic computing control loop adjusts all the systems, the reinforcement learning modules can evaluate applied policies and decide which ones must be used or avoided in the next iteration. Approaches such as those proposed by Vienne and Sourrouille [28] use reinforcement learning to select the rules to be applied for each decision to be taken, having goals such as a performance level for resources. Other works such as those presented by Vengerov and Iakovlev [29, 30] and Perez et al. [31] show frameworks for scheduling resources using reinforcement learning, using as objective function the optimization of the utility of resources. There are also other works such as the one presented by Fenson and Howard [32] that show approaches for self-rejuvenation and self-management functions using these reinforcement learning techniques.

8.2.2.3 Feature and example selection. Feature selection is the process of, given a huge set of observed data, finding those variables and data that are really useful from those that only bring noise or are irrelevant. Usually, we can obtain several data from observing a grid or cloud system. But in order to predict or estimate a specific interesting value, not all the collected data is useful, and including this data in the learning process makes it harder in space or in time, or makes it inaccurate due to noise.

It also happens when applying knowledge to the self-methods, as all the self-management aspects must use the correct information to work: self-healing systems must have the correct signals to detect anomalies and predict the causes and consequences for failures. Self-protection systems must be able to see the indicators about attacks when attackers are cloaked or use evasion techniques. And the self-optimization and self-configuring systems must know about the execution and requirements of applications, as well as utility of the resources in order to find the best configuration and best performance. The feature selection methods are in charge of discovering the relevant attributes from all the data obtained.

Also, there is the example selection process. When finding or training a model, the examples must be "good" examples that cause minimum noise and are less redundant as possible. This example selection process is not trivial, as you want to keep enough examples for your data set to be general enough, or to give enough support to cases hard to be learned. Blum and Langley [33] described in their survey the basics of feature and example selection.

Furthermore, some works on self-management used to perform a principal component analysis (PCA) [34, 35] in order to find the attributes that can differentiate better our examples. PCA is a feature selection technique by itself, but it can also find the combination of features that are most relevant, and is able to treat high dimensional data, reducing the complexity without losing much information. Zheng et al. [36, 37] proposed the utilization of PCA for detecting and locating anomalies in large-scale clusters. In their approaches, after collecting data and finding the combination of attributes that better differentiate the collected examples, an outlier detection is done using the cell-based detection algorithm. Other works such as those by Lakhina et al. [38, 39] use the PCA method to detect anomalies in network traffic, also using classification techniques (supervised learning) to identify network anomalies.

8.3 INTRODUCING POWER-AWARE APPROACHES

At this moment, green computing is being introduced into self-management middleware, adding to these frameworks new advanced management mechanisms to successfully optimize the resource usage to add energy efficiency as one of the fundamental parameters in its management. The current main power-saving techniques applied in the cloud and grid environments are related with virtualization technology, the turning on/off policies, the DVFS, and the hybrid architecture on data centers.

These four technical areas are being covered by approaches that include ad hoc methods, heuristic algorithms, and determined policies, taking into account that this requires experts in the whole system and changes in the system make these approaches to require updates. In order to introduce the *intelligent management*, letting the system to configure and adapt to changes easily, machine learning techniques are starting to be used to improve the previous methods.

8.3.1 Use of Virtualization

Virtualization is one of the key technologies in the cloud that has enabled cost reduction and easier resource management for service providers. As virtualization allows to run several processes, jobs, guest operating system (OS), and also VMs in one or several physical machines or platforms, it makes possible 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 migrated without many handicaps on the base systems, but VMs can also perform optimizations over the host OS and physical machine. The cloud and grid infrastructure take advantage from this technology, decoupling them from the system software of the underlying resource, and letting the movement and migration of VMs in order to place them in the most convenient place.

The main goals of virtualization are to provide a confined environment where applications can be run, limit hardware resource access and usage or expand it transparently for the applications, adapt the runtime environment to the application, use dedicated or optimized OS mechanisms for each application, and manage the whole applications and processes running within VMs. Primet et al. [40] provided a survey on current OS and network virtualization solutions for grids. The summarized basic aspects are listed in the following.

- *OS-Level Approaches*. These approaches allow to virtualize a physical server enabling multiple isolated and secure virtualized servers to run on a single physical server. No guest OS is used, and applications are run in a specific view of the only one OS as if they were running alone on the OS. Some of these approaches are VServer [41], a kernel patch based on partitioning, using a "security context" inside a UNIX OS, FreeBSD Jail [42], and also Solaris Containers, OpenVZ, etc.
- *Emulators*. VMs simulate the complete hardware used by a guest OS. VMware [43] is a virtualization software for machines based on x86 architecture, where virtualization works at the processor level, the VM privileged instructions are trapped and virtualized by the VMware process, and other instructions are directly executed by the host processor. All hardware resources of the machine are also virtualized. Other solutions are Microsoft VirtualPC, VirtualBox, QEMU [44], etc.
- *OS in User Space*. These approaches provide virtualization through the execution of guest OSs directly in user space. Some approaches are User Mode

Linux [45], which allows launching Linux OS as applications of a host machine running Linux, as well as coLinux, Adeos, L4Ka-based projects, etc.

- *Paravirtualization*. The paravirtualization technique does not necessarily simulate the hardware but instead offers a special API requiring modifications to the guest OS. The hardware resources are abstract resources not necessarily similar to the actual hardware resources of the host machine. Xen [46] is a VM monitor for x86 architecture, allowing the concurrent executions of multiple OS while providing resource isolation and execution confinement between them. Other projects using this paravirtualization approach are Denali and Trango.
- *Hardware-Assisted Virtualization*. This virtualization allows to run unmodified guest OS, giving to the VM its own hardware. This is possible thanks to an increased set of processor instructions provided by Intel VT (IVT [47]), AMD (AMD Pacifica x86 virtualization [48]), IBM (IBM Advanced POWER virtualization [49]), and Sun (Sun UltraSPARC T1 hypervisor [50]).

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 past years, works such as the ones presented by Vogels [51] studied the consolidation advantages using virtualization while other works such as the ones from Nathuji et al. [52] have widely explored its advantages from a power efficiency point of view.

Recent work by Petrucci et al. [53] proposed a dynamic configuration approach for power optimization in virtualized server clusters and outlined 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 or off the resources, as it is explained in Section 8.3.2. Also, VM migration and VM placement optimization are studied in the work of Liu et al. [54] to improve the VM placement and consolidate in a better way. On the basis of these works, Goiri et al. [55, 56] introduced the SLA-factor into the self-managing virtualized resource policies. The SLA-driven policies look for facilitating resource management in service providers, allowing cost reduction and at the same time the SLA agreed QoS fulfillment.

So virtualization technology has opened a wide research area to explore in order to optimize cloud and grid management. The capability to isolate jobs inside VMs, and migrate the VMs along physical machines, permits optimizing task placement and dynamic scheduling without much overhead. Recently, machine learning techniques are being applied to help manage virtualized platforms to decide what VMs must be started and how to schedule them, complementing information about the system or predicting useful information a priori. Also, these techniques are able to look for patterns in the behavior of the VMs and host systems to predict their imminent and long-term behavior, making long-term

policies more accurate. These approaches are often applied within turning on and off machines or DVFS, as described in the following section.

8.3.2 Turning On and Off Machines

Another energy-saving technique is to determine when a node should be turned off to save power consumption or when to turn it on to bring service to the cloud. These actions can be driven by fixed policies or heuristics, depending on the load being received at each moment or the load expected in a short-term window time. For example, first approaches such as the ones presented by Pinheiro et al. [57], Chase et al. [58], and Elnozahy et al. [59] applied turning on and off mechanisms for power management, as well as by Chen et al. [60] that includes predictive techniques in a proactive and also reactive automated control.

Goiri et al. [56, 61] showed that a decrease in the number of online machines obviously assures a decrease in the consumed power and also the system is often unable to bring service given an increase in load, so a compromise between online 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. Adequate thresholds can be obtained (this time empirically) in order to decide how many physical machines are needed online, and the rest can be shut down.

On the basis of the same works, Berral et al. [62] proposed a framework that provides an *intelligent* consolidation methodology using different techniques such as 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 used a learned model to predict power consumption levels, CPU loads, and SLA timings and to improve scheduling decisions. The experiments performed using grid workloads and a cloud environment demonstrate how consolidation-aware policies give a better energy efficiency than nonconsolidating ones, and also, the machine learning model responses are much better with respect to power consumption when the information obtained from users and tasks is not uniformly accurate.

This turn-on and off technique is also applied by the approach of Kamitsos et al. [63], which sets unused hosts in a low consumption state to save energy. In their approach a Bellman's function based on 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 in the online hosts.

But turning on and off is not limited to machines. Components and resources can also be started up and shut down. Policies can decide whether to set on or off the full machine or a specific component, and Tan et al. [64] showed a

framework for controlling the system power manager using reinforcement learning algorithms. In this case, the learner uses a Q-learning algorithm, a popular algorithm originally designed to find policies in Markovian decision processes.

Summarizing, the main point in the strategy of turning on and off devices, machines, or resources is to determine at each time what to switch on and off in order to optimize our goals. Optimal policies select at the best time those elements that are necessary for the good performance of the system and maintain the rest in a shutdown or low consumption status. These selection policies can be improved or optimized by reinforcement learning techniques by adjusting the number of elements that are not necessary in the system at each time, and inductive learning techniques can be used to expect the amount of resources to be used a priori in order to plan on/off device schedulers.

8.3.3 Dynamic Voltage and Frequency Scaling

Another currently applied technique to obtain power efficiency is the Dynamic Voltage and Frequency Scaling (DVFS). The DVFS techniques allow the reduction of voltage and frequency, providing substantial saving in power at the cost of slower program execution. Current microprocessors and other kind of resources allow the power management by DVFS, reducing the voltage and frequency of the given devices and allowing the application of policies in order to provide saving in power at the cost of not offering the full capabilities of the resource when not needed. As Chen [60] stated in his work, new power-saving policies, such as DVFS or turning off idle servers, can increase hardware problems and the problem of meeting SLAs in reduced environments. This can be solved by adding a smarter scheduling policy to dynamically turn off idle machines to reduce the overall consumption.

Earlier works on DVFS were mainly focused on power saving on mobile devices while preserving QoS and performance. The first approaches on power management used turning server machines on and off, one of the firsts to combine turning on and off with dynamic voltage scaling in data centers and was studied by Elnozahy et al. [59], exploring the use of DVFS to respond to changes in server demands. This work and the work of Sharma et al. [65] have referred to applying these techniques for server applications, and from here on, other works have developed this idea toward refining and detailing the scheduling procedure in order to decide when and how much voltage and frequency scaling should be applied at each moment.

Reinforcement learning is also used to drive DVFS policies as shown in the works of Tesauro and Kephart et al. [66, 67]. Their goal was to let the system learn the actions to be performed with a trial-and-error method, making decisions by selecting the expected best action and checking the results, allowing to adjust the ranking for the action. In this case, actions control the CPU frequency, adjusting it to the optimal trade-off between electric consumption and response time for transactional jobs running on the given data center.

So tuning the processor voltage and frequency has become an effective method to reduce the power consumption while tasks can be delayed in time, or the

required performance is under the system capabilities. The main challenge here is to know when it is possible to scale up and down voltage/frequency, what policies are optimal to decide when and how much to do so, and what trade-off must be permitted between power consumption and QoS. Current techniques using heuristics and fixed policies are being improved using reinforcement learning methods, finding the optimal policies that assure the lowest power consumption without compromising the performance and service requirements.

8.3.4 Hybrid Nodes and Data Centers

Finally, another power-saving technique is to design data centers and machines as a hybrid architecture, combining high performance and energy-efficient elements, to switch between one another in the order of the load requirements. Turning on and off resources or modifying their consumption, by switching resource usage between the ones designed for energy saving or high performance, could be a good option depending on the situation or requirement of the load. Combining low power designed processors with high performance processors or devices in the same data center provides the system a new degree of freedom, so that there is no need to modify the elements in the system but to use those elements that are more prepared to our energy or performance needs.

This combination of different kind of resources has been tackled in local hosts in some approaches such as the ones presented by Chun et al. [68], who proposed a hybrid architecture that combines the selective usage of processors with different power consumptions and performances in a single host in order to apply local energy-saving policies only when allowed by performance. Also, approaches presented by Nathuji et al. [69] state that a good approach for saving energy is mixing low power systems and high performance ones in the same data center.

Machine learning techniques applied to the utilization of hybrid data centers are still in process, as the current state-of-the-art research applies the knowledge of induction learning works for improving autonomic computing approaches. Again, the works of Goiri et al. [70, 71] have included learned functions [62] as management parameters. Also the techniques presented in Section 8.2, referring to search policies applying reinforcement learning [27, 66], can be applied in order to decide whether to use a determined kind of resource or another. In conclusion, the use of hybrid systems is giving new elements to *intelligent* decision makers, so new solutions are able to optimize, thus regulating the properties of individual elements in order to adapt them to loads and scenarios or instead to decide to select specific elements to fit the specific scenario.

8.4 EXPERIENCES OF APPLYING ML ON POWER-AWARE SELF-MANAGEMENT

After looking at all the works and publications referring to the new techniques improving power-aware self-managed systems using data mining and machine

learning, some experiences on applying them to specific cases of study are shown here. These cases of study refer to the conclusions obtained from some works mentioned before [56, 62], where a cloud is scheduled and managed in a power-aware way, using learning techniques over some of the previously explained power-saving techniques.

8.4.1 Case Study Approach

The techniques proposed for this case study are consolidation techniques, reducing power consumption by scheduling virtualized environments, all without degrading their SLAs in excess. This scheduling policy must consolidate workloads preserving the QoS of the tasks inside a virtual machine (VM) each one, agreed on the SLA and taking into account virtualization overheads such as VM creation, checkpointing, and migration. All of this can be achieved by unifying different provider requirements in addition to power consumption, namely, reliability and dynamic SLA enforcement (be able to recover from an SLA violation during the execution).

This is done by deciding the best location for executing a new job depending on the resources it requires in order to fulfill its SLA, derived from the information of the system, including job execution and node status. The proposed policy periodically calculates whether to move jobs in order to improve global system utility. This approach decides when and where to create VMs containing jobs, migrate them, and start up or shut down physical machines, also being aware that machines in a cluster can have different properties so the data center can be heterogeneous.

In this section, the whole proposed policy is summarized and evaluation and improvements obtained in a first implementation, including virtualization overheads and power consumption, are shown. It is compared against common policies in a simulated environment that models a virtualized data center, mainly focusing in this occasion on CPU and memory as a resource. This first proof of concept is based on HPC jobs and uses deadlines as QoS metric in order to define the SLA constraints. After evaluating the different power-aware techniques, the concept of ML is introduced in order to improve the consolidation mechanisms by predicting information about SLAs before applying the selected schedule.

8.4.2 Scheduling and Power Trade-Off

The scheduling policy consists on finding, on each system status change the optimal combination of <host,VM> to use as input information: the hardware and software requirements of the VM, the amount of resources required, the resources offered by the host machine (i.e., those that are available), the energy consumption of the physical machine, the user SLA constraints, and the reliability of the host. It gives to each machine a dynamic score depending on these parameters and solves the allocation of each VM on the best machine, taking into account all those different factors.

A scheduling round is started when a new VM enters the system, a VM finishes its execution, a violation in the SLA is detected, or the reliability of a machine changes. Then, the best <host,VM> combination is found by mapping all the tentative VM allocations in a scoring matrix filled with the benefit of each VM temptatively hold in each host machine. Each score indicates the utility (or benefit) of holding a VM in a host by aggregating all the penalties related to migration, leaving a machine empty or violating the SLA or any other needed constraint. At this point, a hill-climbing search algorithm [72] finds the set of movements optimizing the <host,VM> matrix. And finally, the system performs the set of operations decided by the new schedule (creation, migration, etc.). Each value in this scoring matrix represents the score (penalization) of hosting a VM in a specific host, including the costs involved due to virtualization, power consumption, reliability, and dynamic SLA enforcement. For example, those hosts that cannot hold a VM, due to insufficient free resources or hardware/software constraints, have an ∞ value; those hosts that have jobs that can be consolidated are penalized in order to force the scheduler to empty it; and the opposite happens with the nearly full hosts that can allocate jobs from nearly empty hosts. In this case, those hosts that have the lowest value for a VM are supposed to be the most suitable.

As it has been already presented, consolidation is applied in order to turn off unneeded machines (also referred as nodes). Nevertheless, a too aggressive node-turning-off policy will result in not offering enough resources to execute tasks, while a passive one will have bigger power consumption. This trade-off depends on the λ_{min} and λ_{max} thresholds. The effect of these two thresholds has been tested by executing the grid workload on top of the simulated data center using the score-based (SB) policy, which is the one that makes a more aggressive consolidation. This allows evaluating the influence of the turning on/off thresholds by showing the client satisfaction and the power consumption, respectively.

Figure 8.3 shows that waiting for the nodes to reach a high utilization before adding new nodes (high λ_{max}) makes the power consumption smaller. In the same manner, the earlier the system shut downs a machine (high λ_{min}), the smaller the power consumption is. It demonstrates how turning on and off machines in a dynamic way can be used to dramatically increase the energy efficiency in a consolidated data center. On the other hand, as shown in Figure 8.4, client satisfaction decreases when the turn on/off mechanism is more aggressive and it shuts down more machines (in order to increase energy efficiency). Therefore, this is a trade-off between the fulfillment of the SLAs and the reduction of the power consumption, whose resolution will eventually depend on the provider interests. For instance, if the provider is having a high client satisfaction, the provider could decide to reduce it slightly while keeping the client between the limits of satisfaction, allowing a greater power reduction by letting resources unused and shutting down them.

Fortunately, average threshold values give a balanced trade-off between energy and QoS. Experimentally, we found that our environment's best values are $\lambda_{min} = 30\%$ and $\lambda_{max} = 90\%$ to ensure almost complete fulfillment of the SLAs while

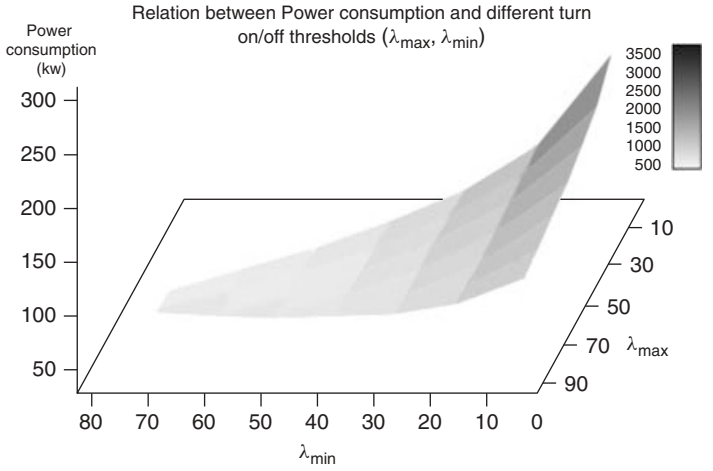

Figure 8.3 Power consumption using different turn on/off thresholds.

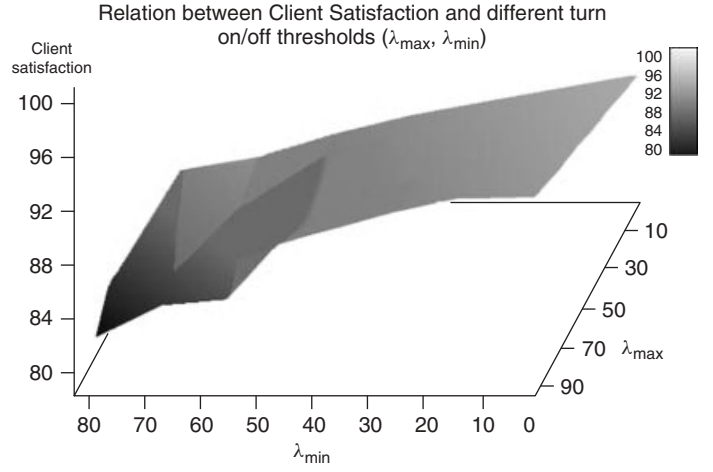

Figure 8.4 Client satisfaction using different turn on/off thresholds.

getting substantial power reduction. A next step would be to dynamically adjust these thresholds, which is part of our current and future work.

8.4.3 Experimenting with Power-Aware Techniques

The experimental environment set for this case study consists of the simulation of a whole virtualized data center with 100 nodes using the EEFSIM, a cloud simulator designed following the procedure in [73] but focused on the energy consumption and on the CPU power scheduling among VMs. The simulator loads

a workload trace, simulates the execution on several machines with different a configuration for each one, and generates the output results using different global scheduling policies. In addition, these machines can be dynamically turned on/off. The simulation takes into account both the physical machine boot time and power consumption and the VM creation and migration power consumption.

The data center is configured to have different types of nodes according to their virtualization overheads, basically different times for creating and migrating VMs. The presented approach intends to take benefit of consolidation in large virtualized data centers executing HPC jobs. For this reason, the evaluation of this case study has been performed using a grid workload, which has been obtained from Grid5000 [74] on the week that starts on Monday, October 1, 2007. The set of policies is evaluated according to different metrics including number of used nodes, CPU usage, power consumption, and SLA fulfillment. The consolidation of the system is reflected in the average number of working nodes (those that are executing a VM), online nodes (those that are turned on) and the power consumption.

Once the parameters to turn nodes on/off have been set up, according to the number of loaded nodes, a first basic experimentation is run. The energy efficiency and SLA fulfillment is compared with four static scheduling algorithms that do not use migration: a random scheduler (RD), which assigns the tasks Random; a Round-Robin scheduler (RR), which assigns a task to each available node; a backfilling strategy (BF), which tries to fill the nodes as much as possible; and a basic version of the presented SB policy (SB0), which just takes into account hardware and software requirements (this time, without migrations).

The results presented in Table 8.1 show the power consumption (Pwr) and different metrics such as the average number of nodes that are actually working (Work), the average number of nodes running (ON), the client satisfaction (S), and the delay. It shows that nonconsolidating policies such as *Random* and *RR* give poor energy efficiency while violating a significant amount of SLAs: they give the worst results on both criteria. *BF* gets better SLA fulfillment with substantially lower cost, as it uses fewer nodes. Finally, the SB policy, which works with no penalties on virtualization overheads, behaves very similar to the *BF* policy.

Using the nonmigrative approach, the SB policy is tested with different configurations ($\text{SB1} = \text{SB0} + P_{\text{virt}}$, $\text{SB2} = \text{SB1} + P_{\text{conc}}$) to test the impact of considering virtualization overheads (creation and concurrency). Table 8.2 shows that SB1, which adds VM creation overheads, makes a better use of the resources

TABLE 8.1 Scheduling Results of Policies Without Migration

	Work/ON	CPU, h	Pwr, kW	S, %	Delay, %
RD	24.3 / 41.7	14597.2	1952.1	33.2	474.5
RR	23.5 / 51.9	11844.2	2321.0	60.4	338.4
BF	10.1 / 22.2	6055.3	1007.3	98.0	10.4
SB0	9.9 / 22.4	6055.3	1016.3	98.2	10.4

TABLE 8.2 Scheduling Results of Score-Based Policies Without Migration

	λ	Work/ON	CPU	Pwr	*S*	Delay
SB0	30–90	9.85/22.4	6055.3	1016.3	98.2	10.4
SB1	30–90	10.2/22.2	6055.3	1006.7	97.9	10.7
SB2	30–90	10.2/23.0	6068.5	1038.5	99.2	8.8
SB2	40–90	10.4/19.0	6055.1	880.5	98.1	10.2

TABLE 8.3 Scheduling Results of Policies With Migration

	λ	Work/ON	CPU	Pwr	*S*	Delay	Mig
DBF	30–90	9.7/21.3	6056.0	970.6	98.1	12.9	124
SB	30–90	9.7/21.0	6055.8	956.4	99.1	9.0	87
SB	40–90	9.7/18.3	6055.8	850.2	98.4	9.9	87

because it takes into account the time to create VMs and selects better nodes to perform the same. In addition, it gets worse by SLA fulfillment than solving it using the SB2, taking care of concurrency overheads, which also causes a small increment on the power consumption. This is because considering the cost of concurrent creation of VMs reduces the consolidation ratio but gets better SLA fulfillment since it produces faster VM creation.

Even though it implies a power consumption increment regarding the basic configuration, the client satisfaction has been increased and allows the provider to make a more aggressive turn on/off policy resulting in higher consolidation and lower power consumption.

Table 8.3 shows the results of the SB scheduler when introducing the capability to migrate VMs in order to get a better consolidation, applying a *dynamic backfilling* (DBF) policy. This applies *BF* and migrates VMs between nodes in order to provide a higher consolidation level and the SB proposal using all the penalties, including the migration capability.

Results for DBF showed a small improvement in power efficiency with respect to nonmigration variation while getting much better consolidation, caused by the overheads introduced by migrating VMs, and the SLA fulfillment is maintained at a medium level as in the nonmigration approach. On the other hand, the SB policy takes virtualization overheads such as creation and migration into account, which makes it to get more client satisfaction. And as in the previous experiment, to give a measure of the improvement in client satisfaction terms, a similar SLA fulfillment target for DBF and the best of the SB configurations are set, getting more aggressive turn on/off parameters of $\lambda_{min} = 40\%$ and $\lambda_{max} = 90\%$. Using this configuration, a reduction in the data-center power consumption of 15% with regard to *BF* is obtained and of 12% when compared with the dynamic variant. These experiments demonstrate how the SB proposal gets the best power consumption and SLA fulfillment, as it takes into account the migration overheads.

TABLE 8.4 Score-Based Scheduling Results With Different Costs

C_e	C_f	Work/ON	CPU	Pwr	S	Delay	Mig
0	40	10.4/22.9	6055.2	1036.4	99.3	8.6	0
20	40	9.7/21.0	6055.8	956.4	99.1	9.0	87
60	100	9.3/22.0	6057.8	998.8	97.7	11.2	432

One of the advantages of the power-aware SB policy is that it can be easily configured according to the provider's requirements. In this experiment, some variants of the policy are shown: without penalizing empty hosts, using typical parameters penalizing empty hosts and rewarding near full host, and using more aggressive parameters for consolidation.

Table 8.4 shows the results of tweaking this parameter. The first variant does not penalize empty hosts, which implies lower consolidation and worst power performance, and also does not migrate any VMs since the fillable reward is not worthwhile. The second variant uses the values used in the previous experiments, which include the empty host penalization and gets better consolidation while maintaining similar client satisfaction as it performs an accurate number of migrations. Finally, the third variant has been set up with aggressive parameters, getting the best consolidation in terms of working nodes, but getting poor energy efficiency and lower SLA fulfillment, which is mainly because it rewards the occupation and penalties too much empty hosts, which implies a big amount of migrations.

8.4.4 Applying Machine Learning

Applying consolidation mechanisms such as the dynamic backfilling described earlier helps to improve power consumption, but often this can be improved or easily done by applying knowledge-based techniques such as machine learning. A first approach focused on learning about the behavior of a job being placed in a specific target physical machine is discussed here.

The machine learning-aided policy implements a dynamic BF scheduler, using the information provided directly by the user and using as decision maker the results of performance and power consumption estimators. When having a pair <host,VM> in the schedule, the impact the job will cause in the potential host machine is predicted using context information. So the scheduler has better confidence on the fact that the selected jobs schedule will not degrade their performance violating SLAs.

When a new job arrives, the system will try to allocate it, so the candidate moves will be like "move VM v from its initial temporary host to host h". Performing a sample run the scheduler can obtain for each combination of VMs and hosts the performance result of this combination, joint with the information of the VM (size, requirements, resources used) and the information of the host (capacity, resources available, information about the other jobs running on it).

From here on, using machine learning algorithms, a model can be set up, learning the relation of these elements with the response variable (the resulting performance of the job). This model will help the decision maker in predicting values of expected performance for a given combination before the schedule is applied.

Running some experimentation over the presented scheduling problem the behavior and performance of different scheduling policies are evaluated using three different workloads (an HPC, a transactional workload, and an heterogeneous one) and using the turn on/off thresholds $\lambda_{min} = 30\%$ and $\lambda_{max} = 60\%$. Also, some scheduling algorithms are evaluated for comparing them with the power-aware one: Random and RR do not use any user-provided information about the VMs and do not consolidate. For BF and DBF, the user provides for each VM a figure indicating which percentage of a CPU capacity should suffice to satisfy the VM SLAs. The algorithms trust this figure as totally reliable and therefore will make decisions that may fit the SLAs very tightly, thus saving power. The algorithm applying machine learning does not use the user-provided information but only uses information about the online requirements of the VM in order to expect the SLA future performance. The results are presented in Table 8.5.

The results obtained using the grid workload show that nonconsolidating policies such as Random and RR give a poor energy efficiency while violating some SLAs. BF and DBF fulfill all SLAs with substantially lower cost, and machine learning performs almost perfectly with respect to SLAs (as we have seen that

TABLE 8.5 Scheduling Results

	$\overline{\text{Work}}$	$\overline{\text{ON}}$	CPU usage, h	Power, kW	SLA, %
Grid workload					
RD	16.51	40.76	6017.85	1671.16	88.38
RR	16.11	41.37	5954.91	1696.66	85.99
BF	10.18	27.10	6022.34	1141.65	100.00
DBF	9.91	26.46	6104.33	1118.86	100.00
Machine learning DBF	15.04	37.92	6022.27	1574.78	99.69
Service workload					
RD	218.46	400.00	75336.88	19784.38	100.00
RR	290.99	400.00	78419.97	19761.54	100.00
BF	108.79	352.88	59792.09	16257.26	100.00
DBF	108.79	352.88	59748.10	16229.22	100.00
Machine learning DBF	99.61	270.50	61379.38	13673.71	100.00
Heterogeneous workload					
RD	224.08	400.00	82137.27	19763.63	88.53
RR	260.66	400.00	84432.96	19713.72	94.20
BF	110.85	330.19	65894.46	16304.38	99.50
DBF	111.03	329.07	66020.58	16214.49	99.59
Machine learning DBF	124.20	307.89	68554.01	15110.33	98.63

predictions for SLA fulfillment are very accurate). The reason is that the user-provided figures for the tasks are very close to the real ones (and the load quite steady), so the BF algorithms will take many decisions that will not violate any SLA but that look too risky to machine learning, which pays a high price in consumption for its caution.

On the service workload, the machine learning scheduler is much better with respect to energy consumption. Note that in this workload, all the schedulers executed all the tasks, so all SLAs were fulfilled. The workload has a very variable CPU usage. This means that the user-provided estimation about the CPU to be used for the given jobs will be a large overestimation for large periods (while it was very tight on the grid workload), and power will be unnecessarily wasted. The machine learning scheduler, as being more conservative, estimates better the SLA fulfillment, and so it is able to reduce the power consumption just to the required.

Finally, the results obtained using the heterogeneous workload are, as expected, a mix of the two previous workloads. In this case, the overall SLA fulfillment by the ML is worse by about 1%, but its overall power consumption is better by about 10%.

8.4.5 Conclusions from the Experiments

These works and experimentations applying power-aware mechanisms introducing machine learning are looking to provide a vertical and intelligent consolidation methodology to deal with uncertain information, keeping in mind performance and power consumption. And the results obtained indicated that significant improvements can be achieved using machine learning models to predict schedules a priori and decide the movements and operations to be done within scheduling functions.

The experiments using the grid workload demonstrate how non-consolidation policies result in poor energy efficiency compared to consolidation-aware policies such as the BF and scoring policies. Also, the machine learning method is close enough to these models that use external information with respect to SLA fulfillment (performance) and much better with respect to power consumption when the information provided by the users is not uniformly accurate, or the information is more variable.

8.5 CONCLUSIONS ON INTELLIGENT POWER-AWARE SELF-MANAGEMENT

As discussed in this chapter, data-center power-aware management techniques are mainly focused on the autonomic computing field, so power optimization is done automatically by middleware software in order to deal with the big growth of the IT infrastructure and the cloud. Furthermore, this automated control is not just having refined policies, as the systems usually change, requiring the need of

self-adaption. While autonomic computing properties have several techniques to update their status, the most useful and currently applied techniques are data mining and machine learning. These learning methods not only can model the system with good accuracy from examples but also let adapt the model to changes easily.

From the four power-saving strategies presented in this chapter (virtualization, turning on/off machines, DVFS, and hybrid architecture), recent approaches have started to apply knowledge-based systems, improving the techniques or allowing to apply them all together. Instance-based learning can help to complement inexistent or uncertain data when dealing with observed data, and also, it can find new data or discover hidden variables from the system that can be relevant for determining the behavior of the resources, the clients, or any other element of the cloud or data center. Furthermore, reinforcement learning techniques are being applied to decide changes in the system policies, so at each step in the system loop the learner can observe how good was the previously applied action and then prioritize again the actions and modify its policies. Also feature and example selection, with techniques such as PCA, are being applied to identify which information obtained from the system is useful, erase outliers and not correct examples, and find the important attributes that influence most of the system.

In the case of virtualization, most machine learning techniques are dedicated to predict the system status before and after each creation, migration, or modification of VMs. Estimating a priori the benefit of realizing a VM operation can reduce the number of useless operations or drawbacks after operating. Consolidation is applied as the principal technique when virtualizing, so estimating and predicting the correct level of consolidation helps to find the optimal power-saving schedule in the system.

The policies based on turning machines on and off tend to apply reinforcement learning and dynamic programming formulas, mainly because deciding when to turn needed or unused machines on or off is an *easy* policy to learn from executions. The same happens with DVFS, where resources and devices are regulated using reinforcement learning too, so finding good policies in order to adjust levels of power or processor frequency in an optimal way can be achieved by trial and error during executions and is also a very adaptive technique.

Finally, the construction and usage of hybrid architecture allow the manager to decide what kind of resources to use in each moment. If self-adaptive management is applied using techniques such as load prediction, RL learned policies and a priori data obtainment, the decision maker has some help to find the most suitable resource scheduling, evaluating not only the load but also the kind of resources to be used, always using power consumption as one of the most important parameters.

To conclude this chapter, there are many works applying knowledge and learning techniques in self-management, but there is more to do so that traditional decision makers can evolve into new ones that are able to detect the relevant information to describe a system, adapt the decision rules when the system changes or when new elements enter into it, or use the experience and learn to predict future states of the system and act in consequence. There are several useful works on

machine learning and data mining awaiting to be applied in cloud and data-center management situations, and there are many works in self-management awaiting to be improved and upgraded using new knowledge and learning methods.

REFERENCES

1. Koomey J. Estimating regional power consumption by servers: A technical note; 2007.
2. Vraalsen F. Performance contracts: Predicting and monitoring grid application behavior; 2001.
3. Fahringer T. Automatic performance prediction of parallel programs; 1996.
4. Deelman E, Blythe J, Gil Y, Kesselman C. Mapping abstract complex workflows onto grid environments; 2004.
5. Gil Y, Deelman E, Blythe J, Kesselman C, Tangmunarunkit H. Artificial intelligence and grids: workflow planning and beyond; 2004.
6. Shadbolt NR De Roure D, Jennings NR. The semantic grid: past, present, and future; 2005.
7. Ejarque J, de Palol M, Goiri I, Juliã F, Guitart J, Badia R, Torres J. Exploiting semantics and virtualization for sla-driven resource allocation in service providers; 2010.
8. Wolpert DH, Wheeler KR, Tumer K. Collective intelligence for control of distributed dynamical systems; 1999.
9. Sirlantzis K, Fairhurst MC, Hoque S. Genetic algorithms for multi-classifier system configuration: A case study in character recognition. In: MCS '01: Proceedings of the 2nd International Workshop on Multiple Classifier Systems. London: Springer-Verlag; 2001. pp. 99–108.
10. Rahman AFR, Fairhurst MC, Hoque S. Novel approaches to optimized self-configuration in high performance multiple-expert classifiers. In: IWFHR '02: Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR'02). Washington (DC): IEEE Computer Society; 2002. p. 189.
11. Kapadia NH, Fortes JAB, Brodley CE. Predictive application-performance modeling in a computational grid environment; 1999.
12. Hofer J, Fahringer T. Grid application fault diagnosis using wrapper services and machine learning. In: ICSOC '07: Proceedings of the 5th International Conference on Service-Oriented Computing. Berlin, Heidelberg: Springer-Verlag; 2007. pp. 233–244.
13. Zhang Q, Cherkasova L, Mi N, Smirni E. A regression-based analytic model for capacity planning of multi-tier applications. Cluster Comput 2008;11(3):197–211.
14. Alonso J, Torres J, Silva LM, Griffith R, Kaiser G. Towards self-adaptable monitoring framework for self-healing. Technical Report TR-0150, Institute on Architectural issues: scalability, dependability, adaptability, CoreGRID - Network of Excellence; 2008 July.
15. Alonso J, Berral JL, Gavaldà R, Torres J. Predicting web application crashes using machine learning. In: Submitted to ACM/IFIP/USENIX 10th International Middleware Conference; 2009 Nov; Urbana Champaign (IL).

16. Alonso J, Torres J, Gavalda R. Predicting web server crashes: A case study in comparing prediction algorithms. In: ICAS 2009: Proceedings of the International Conference on Autonomous Systems; April 20–25, 2009. Valencia, Spain.
17. Poggi N, Moreno T, Berral JL, Gavaldà R, Torres J. Web customer modeling for automated session prioritization on high traffic sites. In: UM '07: Proceedings of the 11th International Conference on User Modeling. Berlin, Heidelberg: Springer-Verlag; 2007. pp. 450–454.
18. Moreno T, Poggi N, Berral JL, Gavaldà R, Torres J. Policy-based autonomous bidding for overload management in ecommerce websites. In: Proceedings of the Group Decision and Negotiation 2007. Springer-Verlag; Montreal, Canada; 14-1 of May, 2007; pp. 162–166.
19. Poggi N, Moreno T, Berral JL, Gavaldí R, Torres J. Self-adaptive utility-based web session management. Comput Netw 2009;53(10):1712–1721.
20. Witten Ian H., Frank E., Hall Mark A.: Data Mining: Practical Machine Learning Tools and Techniques (Third Edition). Morgan Kaufmann; January 2011; 629 pages; ISBN 978-0-12-374856-0.
21. Wildstrom J, Witchel E, Mooney RJ. Towards self-configuring hardware for distributed computer systems. In: ICAC '05: Proceedings of the 2nd International Conference on Automatic Computing. Washington (DC): IEEE Computer Society; 2005. pp. 241–249.
22. Wildstrom J, Stone P, Witchel E, Dahlin M. Machine learning for on-line hardware reconfiguration. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence; Hyderabad, India; 2007. pp. 1113–1118.
23. Wildstrom J, Stone P, Witchel E. Autonomous return on investment analysis of additional processing resources. In: ICAC '07: Proceedings of the 4th International Conference on Autonomic Computing. Jacksonville, Florida, USA; June 11–15, 2007; IEEE Computer Society; 2007. p. 15.
24. Kaelbling LP, Littman ML, Moore AW. Reinforcement learning: A survey. J Artif Intell Res 1996;4:237–285.
25. Sutton RS, Barto AG. Reinforcement learning: an introduction (Adaptive Computation and Machine Learning). The MIT Press; Massachusetts, USA; 1998.
26. Bertsekas DP, Tsitsiklis JN. Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3). Athena Scientific; 1996 May.
27. Tesauro G, Jong NK, Das R, Bennani MN. On the use of hybrid reinforcement learning for autonomic resource allocation. Cluster Comput 2007;10(3):287–299.
28. Vienne P, Sourrouille J-L. A middleware for autonomic qos management based on learning; 2005.
29. Vengerov D, Iakovlev N. A reinforcement learning framework for dynamic resource allocation: First results. In: ICAC '05: Proceedings of the 2nd International Conference on Automatic Computing. Washington (DC): IEEE Computer Society; 2005. pp. 339–340.
30. Vengerov D. A reinforcement learning framework for online data migration in hierarchical storage systems. J Supercomput 2008;43(1):1–19.
31. Perez J, Germain-Renaud C, Kégl B, Loomis C. Utility-based reinforcement learning for reactive grids. In: ICAC '08: Proceedings of the 2008 International Conference on Autonomic Computing. Washington (DC): IEEE Computer Society; 2008. pp. 205–206.

32. Fenson E, Howard R. Reinforcement learning for autonomic network repair. In: ICAC '04: Proceedings of the 1st International Conference on Autonomic Computing. Washington (DC): IEEE Computer Society; 2004. pp. 284–285.

33. Blum AL, Langley P. Selection of relevant features and examples in machine learning. Artif Intell 1997;97(1–2):245–271.

34. Duda RO, Hart PE, Stork DG. *Pattern Classification*. 2nd ed. Wiley-Interscience; New York, USA; 2000.

35. Smith LI. A tutorial on principal components analysis; 2002.

36. Zheng AX, Lloyd J, Brewer E. Failure diagnosis using decision trees. In: ICAC '04: Proceedings of the 1st International Conference on Autonomic Computing. Washington (DC): IEEE Computer Society; 2004. pp. 36–43.

37. Zheng Z, Li Y, Lan Z. Anomaly localization in large-scale clusters. In: 2007 IEEE International Conference on Cluster Computing; Texas, USA; 2007 Sept. pp. 322–330.

38. Lakhina A, Crovella M, Diot C. Mining anomalies using traffic feature distributions. SIGCOMM Comput Commun Rev 2005;35(4):217–228.

39. Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies; 2004.

40. Vicat-Blanc Primet P, Gelas J-P, Mornard O, Mon Divakaran D, Bozonnet P, Jan M, Roca V, Giraud L. State of the art of os and network virtualization solutions for grids. Technical report, INRIA, September 2007. "Delivrable #1: HIPCAL ANR-06-CIS-005".

41. Linux Vserver. Linux vserver. Available at http://linux-vserver.org/Paper; April 2012.

42. The FreeBSD Project. The freebsd documentation project; 2007.

43. VMware Inc. Vmware. Available at http://www.vmware.com/; April 2012.

44. Bellard F. Qemu, a fast and portable dynamic translator. In: ATEC '05: Proceedings of the Annual Conference on USENIX Annual Technical Conference. Berkeley (CA): USENIX Association; 2005. pp. 41–41.

45. Hoxer H, Buchacker K, Sieh V. Implementing a user mode linux with minimal changes from original kernel. In: Proceedings of the 2002 International Linux System Technology Conference; Cologne, Germany; 2002. pp. 72–82.

46. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. In: SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles. New York: ACM; 2003. pp. 164–177.

47. Intel. Intel virtualization technologies. Available at http://www.intel.com/technology/virtualization/; April 2012.

48. AMD. Pacifica x86 virtualization. Available at http://enterprise.amd.com/us-en/AMD-Business/Business-Solutions/Consolidation/Virtualization.aspx; April 2012.

49. IBM. IBM advanced power virtualization. Available at http://www-03.ibm.com/systems/p/apv/f; April 2012.

50. Sun Microsystems. Sun ultrasparc t1 hypervisor. Available at http://opensparc-t1.sunsource.net/specs/Hypervisor-api-current-draft.pdf; April 2012.

51. Vogels W. Beyond server consolidation. Queue 2008;6(1):20–26.

52. Nathuji R, Schwan K, Somani A, Joshi Y. Vpm tokens: virtual machine-aware power budgeting in datacenters. Cluster Comput 2009;12(2):189–203.

53. Petrucci V, Loques O, Mossé D. A dynamic configuration model for power-efficient virtualized server clusters. In: 11th Brazillian Workshop on Real-Time and Embedded Systems (WTR); 2009 May 25; Recife, Brazil.

54. Liu L, Wang H, Liu X, Jin X, He WB, Wang QB, Chen Y. GreenCloud: a new architecture for green data center. In: 6th International Conference on Autonomic Computing and Communications, Industry Session; 2009 June 15–19. Barcelona, Spain: ACM; 2009. pp. 29–38.

55. Goiri I, Julià F, Ejarque J, De Palol M, Badia RM, Guitart J, Torres J. 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); 2009 July 9–11; Cambridge (MA). pp. 211–218.

56. Goiri I, Julià F, Nou R, Berral J, Guitart J, Torres J. Energy-aware scheduling in virtualized datacenters. In: Proceedings of the 12th IEEE International Conference on Cluster Computing (Cluster 2010); 2010 Sept 20–24; Heraklion, Crete, Greece.

57. Pinheiro E, Bianchini R, Carrera EV, Heath T. Load balancing and unbalancing for power and performance in cluster-based systems. In: Workshop on Compilers and Operating Systems for Low Power, Volume 180. Citeseer; 2001. pp. 182–195.

58. Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP. Managing energy and server resources in hosting centers. SIGOPS Oper Syst Rev 2001;35(5):103–116.

59. Elnozahy E, Kistler M, Rajamony R. Energy-efficient server clusters. In: Falsafi B, Vijaykumar T, editors. Volume 2325, Power-Aware Computer Systems, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer; 2003. pp. 179–197.

60. Chen Y, Das A, Qin W, Sivasubramaniam A, Wang Q, Gautam N. Managing server energy and operational costs in hosting centers. ACM SIGMETRICS Perform Eval Rev 2005;33(1):303–314.

61. Fitó JO, Goiri I, Guitart J. SLA-driven elastic cloud hosting provider. In: Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP'10); 2010 Feb 17–19; Pisa, Italy. pp. 111–118.

62. Berral J, Goiri I, Nou R, Julià F, Guitart J, Gavalda R, Torres J. 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; 2010 Apr 13–15 Germany; 2010. pp. 215–224.

63. Kamitsos I, Andrew L, Kim H, Chiang M. Optimal sleep patterns for serving delay-tolerant jobs. In: 1st International Conference on Energy-Efficient Computing and Networking (eEnergy'10), University of Passau; 2010 Apr 13–15; Germany.

64. Tan Y, Liu W, Qiu Q. Adaptive power management using reinforcement learning. In: ICCAD '09: Proceedings of the 2009 International Conference on Computer-Aided Design. New York: ACM; 2009. pp. 461–467.

65. Sharma V, Thomas A, Abdelzaher T, Skadron K, Lu Z. Power-aware qos management in web servers. In: RTSS '03: Proceedings of the 24th IEEE International Real-Time Systems Symposium. Washington (DC): IEEE Computer Society; 2003. p. 63.

66. Tesauro G, Das R, Chan H, Kephart JO, Lefurgy C, Levine DW, Rawson F. Managing power consumption and performance of computing systems using reinforcement learning. In: Advances in Neural Information Processing Systems 20; Vancouver, Canada; 2008.

67. Kephart JO, Chan H, Das R, Levine DW, Tesauro G, Rawson F, Lefurgy C. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In: ICAC '07: Proceedings of the 4th International Conference on Autonomic Computing. Washington (DC): IEEE Computer Society; 2007. p. 24.

68. Chun B, Iannaccone G, Iannaccone G, Katz R, Gunho L, Niccolini L. An energy case for hybrid datacenters. In: Workshop on Power Aware Computing and Systems (HotPower'09); 2009 Oct 10; Big Sky (MT).

69. Nathuji R, Isci C, Gorbatov E. Exploiting platform heterogeneity for power efficient data centers. In: Proceedings of the IEEE International Conference on Autonomic Computing (ICAC'07); 2007 Jun 11–15; Jacksonville (FL).

70. Goiri I, Fitó O, Julià F, Nou R, Berral J, Guitart J, Torres J. Multifaceted Resource Management for Dealing with Heterogeneous Workloads in Virtualized Data Centers. In: Proceedings of the 11th ACM/IEEE International Conference on Grid Computing (Grid 2010); 2010 Oct 25–29; Brussels, Belgium.

71. Goiri I, Guitart J, Torres J. Characterizing cloud federation for enhancing Providers' profit. In: Proceedings of the 3rd International conference on Cloud Computing (CLOUD 2010); 2010 Jul 5–10; Miami (FL). pp. 123–130.

72. Russell S, Norvig P. Artificial intelligence: a modern approach. Pearson Education; 2003.

73. Nou R, Kounev S, Julià F, Torres J. Autonomic QoS Control in Enterprise Grid Environments using Online Simulation. J Syst Softw 2009;82(3):486–502.

74. The Grid Workloads Archive. 2009. Available at http://gwa.ewi.tudelft.nl; April 2012.