Enhancing Cloud energy models for optimizing datacenters efficiency

Edouard Outin, Jean-Emile Dartois
b-com
Rennes, France
Email: firstname.name@b-com.com

Oliver Barais
b-com, IRISA, INRIA, Université de Rennes 1
Rennes, France
Email: barais@irisa.com

Jean-Louis Pazat
b-com, IRISA, INRIA, INSA
Rennes, France
Email: Jean-Louis.Pazat@irisa.fr

***Abstract*—Due to high electricity consumption in the Cloud datacenters, providers aim at maximizing energy efficiency through VM consolidation, accurate resource allocation or adjusting VM usage. More generally, the provider attempts to optimize resource utilization. However, while minimizing expenses, the Cloud operator still needs to conform to SLA constraints negotiated with customers (such as latency, downtime, affinity, placement, response time or duplication). Consequently, optimizing a Cloud configuration is a multi-objective problem. As a nontrivial multi-objective optimization problem, there does not exist a single solution that simultaneously optimizes each objective. There exists a (possibly infinite) number of Pareto optimal solutions. Evolutionary algorithms are popular approaches for generating Pareto optimal solutions to a multi-objective optimization problem. Most of these solutions use a fitness function to assess the quality of the candidates. However, regarding the energy consumption estimation, the fitness function can be approximative and lead to some imprecisions compared to the real observed data. This paper presents a system that uses a genetic algorithm to optimize Cloud energy consumption and machine learning techniques to improve the fitness function regarding a real distributed cluster of server. We have carried out experiments on the OpenStack platform to validate our solution. This experimentation shows that the machine learning produces an accurate energy model, predicting precise values for the simulation.**

***Keywords*-Cloud computing; energy efficiency; model;**

I. Introduction

Simulations are becoming increasingly popular in the Cloud computing field [11], [18]. They are used for various purposes, from modeling to experimentation: checking system behavior or validating optimization algorithm. Moreover, they allow to run reproducible experiments and simulations in a controllable environment, using manageable methodologies for comparing VM placement algorithms or evaluation of algorithms, applications, and policies before actual development on the real infrastructure.

Additionally, simulations and models can also be embedded at runtime [3]. It can be useful for the decision making, checking a configuration validity or testing an optimization algorithm before applying it to the system. Models and simulations are also widely used through metaheuristics to solve multi-objectives optimizations. A possible solution is instantiated as a model, and the optimization algorithm can perform computations on this model to evaluate and recombine it.

For instance, the genetic algorithm mimics population evolution to produce solutions and each candidate is an abstraction of a viable organization of the Cloud infrastructure exprimed as an instance of the model. The solutions are evaluated with their fitness function to find the best trade-off between constraints and goals (e.g. energy-efficiency, service level agreement).

However, the quality of the fitness function depends on the underlying model; if the model is approximate or rough, the algorithm can be biased due to the inaccurate fitness function evaluation leading to suboptimal adaptations. In the context of energy consumption optimization, this raises two research questions we address in this paper.

- **RQ1**. Do differences exist between the energy simulation based on hardware specifications and the real data that can be observed?
- **RQ2**. Could we use machine learning techniques at runtime to improve the simulation accuracy?

In the following paper, we answer these two following research question through experimentation.

This paper is organized as follows. Section II reviews the related work in this field of Cloud computing, simulation and energy modeling and highlights the gap between energy simulation based on hardware specifications and reality. This section details our experimental protocol used for benchmarking the energy consumption under different workloads. Section III presents our approach that combined search-based techniques with learning techniques to improve the fitness function accuracy. Section IV validates our approach through an experimental protocol that answers to the three research questions. Finally, section VI concludes this experiments and discusses questions raised by this experiment.

II. Problem statement

A. Existing energy-aware simulators

Energy-efficiency in Cloud computing environments is an active research topic [25], [13]. Modeling the energy consumption of a server or a whole datacenter is not trivial [22],

/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCAC.2015.10

and simulators keep trying to get as close as possible to the reality. Several projects are widely used in the research community and the industry [10].

CloudSim [11] is developed at the CLOUDS laboratory of the University of Melbourne. Its goal is "to provide a generalized and extensible simulation framework that enables modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services". Indeed, it allows its users to focus on system design issues without having to worry about the low level details inherent to Cloud-based infrastructures and services. It has support for modeling and simulation of energy-aware computational resources. CloudSim energy model is based on the host CPU utilization. For a given utilization on a given host model they get a corresponding energy consumption in Watts. All those values are issued from the *spec.org* website, containing benchmarks for various server models. The Standard Performance Evaluation Corporation (SPEC) is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers.

GreenCloud simulator [18] has been elaborated at the University of Luxembourg. It is a packet level simulator that uses the existing Network Simulator 2 [4] libraries with a strong emphasis on networking and energy awareness. It allows the tracking of the energy consumed by the different components of a Cloud computing environment. Moreover, GreenCloud models the various entities of the Cloud such as servers, switches, links for communication, and the energy they consume. This simulator has independent energy models for each type of resource (e.g. CPU, RAM, disk, network). Nevertheless, the base values and the different coefficients pondering the components consumptions are set manually, they are subject to change from a datacenter to another due to the hardware heterogeneity. Determining those values and coefficients remains the complex part and it can not be approximated.

SimGrid [12] is a scientific instrument developed jointly by the University of Hawaii at Manoa, the LIG laboratory and the University of Nancy. It allows to study the behavior of large-scale distributed systems such as Grids, Clouds, HPC or P2P systems through simulation. SimGrid can be used to evaluate heuristics or prototype applications.
Originally, the simulator does not have any energy awareness. Yet, with the increasing concern on energy-efficiency, the SURF Energy Plugin has been introduced in SimGrid, enabling to account not only for computation time, but also for the dissipated energy in the simulated platform. Nevertheless, this energy model only takes CPU into consideration, it does not deal with other components such as disk or memory. Moreover, it supposes that energy consumption is linear with the CPU utilization. Here is how to declare a host energy model in SimGrid:

```
<host id="HostA" power="100.0Mf">
    <prop id="watt_per_state" value="100.0:200.0"/>
    <prop id="watt_off" value="10"/>
</host>
```

Listing 1. SimGrid host energy model

This configuration applies for a given host, having a given performance exprimed in FLOPS. The first property means that when the host is up and running, but without any CPU utilization, it will dissipate 100 Watts. If it's fully loaded, it will dissipate 200 Watts and if its load is at 50%, then it will dissipate 150 Watts. The second property means that when the host is turned off, it will dissipate only 10 Watts. It is trivial to see that this energy model is too rough as it seems restrictive to express a host consumption using a linear function between two points.

iCanCloud [24] is a simulation platform that aims to model and simulate Cloud computing systems. It has been developed by research group ARCOS at the University Carols III of Madrid, Spain. This simulator can predict the trade-offs between cost and performance of a given set of applications executed in a specific hardware, and then provide to users useful information about such costs. It also has support for modeling precisely the hardware energy consumption of a system such as CPUs, memories, disks, PSUs. For instance, the CPU has several power states like C0, C1 and C3. The C0 is the operating state and C3 is a state where the processor is sleeping. The states of the CPU are associate to the energy consumption in Watts.

The workloads of iCanCloud are based on predefined collections of applications that can be customized. These applications are used to configure tasks that will be performed. However, a Cloud infrastructure is used by various customers and it is not trivial to predetermine the wide scope of applications to simulate.

All these simulators can be used in a classical analysis step of a MAPE-K loop to self-adapt a set of VMs on a Cloud infrastructure. Usually, this analysis step is done using hard coded "static" rules, also called Event-Condition-Action (ECA) engines [21]. Here, we plan to use and manipulate simulators instead of the ECA engine. Each of these simulators has its own simulation models that can be parameterized depending on a real Cloud infrastructure. This parameterization can be done using the public hardware specifications from *spec.org*.

B. Comparing simulator values with measurements

We conduct an experiment to evaluate the simulation accuracy of simulators based on the *spec.org* values for the DELL PowerEdge R620 server to answer RQ1.

1) Experimental protocol: We compare the simulations results and the experimental results we got from the most popular simulator. To select it, we use Google Scholar metrics to identify the most cited simulator. CloudSim has more than 1100 citations. We also select this simulator as it is a popular tool used for modeling and solving Cloud optimization problems [23].

Using stress-ng [7] (enhanced version of stress [8]) we generate some variable load (first in terms of CPU and then, coupled with RAM and disk operations). Additionally, we set up a power distribution unit able to determine the energy consumed by a server and we request the PDU metrics for this server through SNMP. Energy consumption information can be retrieved from the ACPI embedded in the server. Nevertheless, we do not use this mechanism but we use an external observer for the energy consumption, based on a PDU Raritan Dominion[1]. We noticed some differences between the values from the PDU and the ones from the DELL iDRAC: around 10 Watts. This difference can be explained as the iDRAC card is not very sensitive and it does not take into account the whole energy consumed.

Our experiment protocol is the following:

- On the server, we run stress tools to mimic variable server utilization.
- PDU keeps monitoring the server energy consumption.
- Every second, we retrieve information from the server and the PDU.

We conducted two distinct experiments to confront the values. For each experiment, we installed a fresh Ubuntu Server 14.04.2 LTS on the DELL PowerEdge R620, and plugged its power cable on the PDU Raritan Dominion PX outlet. The PowerEdge R620 has 40 logic cores (Intel Xeon E5-2660 - 2.2 GHz), 132 Go RAM, 357 Go HDD disk and runs the 3.13.0-24-generic x86_64 GNU/Linux kernel. All the server metrics measurements are done using *psutil 2.2.1* Python 2.7 library. Below, the details of the two different setups and their respective results.

2) Bare metal: For the first experiment, we do not have any hypervisor. We directly stress the host operating system.

The experimental protocol is the following. We stress the CPU for a range of given utilization level (from 0 to 100, with step of 10). Note that, for some workloads, the CPU load starts at 10% due to the intensive disk write or memory filling. For each step, we have a 120 seconds interval, and we compute the average energy consumption for this duration.

On the figure 1, the blue line corresponds to the *spec.org* values [6]. The green line represents the evolution of the consumption while stressing CPU utilization; the light blue one, while stressing RAM and CPU; and the red one, with disk operations. As we can see on the line graph, there is a gap between the measured values and the ones used by CloudSim.

[1]http://www.raritan.com/products/power-distribution

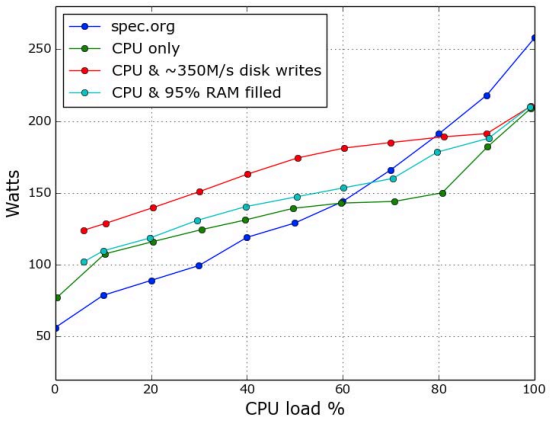

Figure 1. DELL PowerEdge R620 benchmark - No hypervisor

3) Hypervisor and VM: For the second experiment, we installed the KVM hypervisor on the server and created a large Ubuntu Server 14.04.2 virtual machine with a flavor occupying total server CPU and RAM available (40 logic cores, 132 Go RAM, 80 Go disk). Moreover, we installed libvirt-bin to ease the deployment and management of the virtual machines on the server. Then, we stress the virtual machine CPU for a range of given utilization level (from 0 to 100, with step of 10). Note that, for some workloads, the CPU load starts at 10% due to the intensive disk write or memory filling. For each step, we have a 120 seconds interval, and we compute the average energy consumption for this duration. On the figure 2, the blue line corresponds to the *spec.org* values [6]. The green line represents the evolution of the consumption while stressing CPU utilization; the light blue one, while stressing RAM and CPU; and the red one, with disk operations.

At the idle time (0-1% stress CPU), we can notice a non-negligible gap between the value from *spec.org* and the value effectively measured in our experiment. We think it could be the overhead due to the KVM hypervisor [27]. Moreover, we can note that for lower utilizations levels, *spec.org* tends to minimize the energy consumed by the system. Whereas for higher utilizations, it tends to be maximized.
As we expected, the workloads with additional RAM usage (light blue line) and disk operations (red line) have an energy consumption overhead compared to the base workload, stressing only CPU.

The difference is higher for lower utilization levels, in comparison with the first experiment.

4) Results: These experiments permit to answer to $RQ1$. They illustrate that the CloudSim simulation values (based

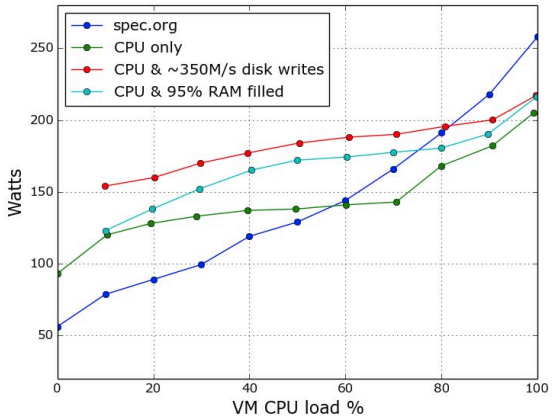

Figure 2. DELL PowerEdge R620 benchmark - Hypervisor

on the *spec.org* data) are not very accurate, it does not reflect the reality of the energy consumption of the system - despite having the global trend. Moreover, it also shows that we can not fully rely on the CPU metric to predict the Watts consumed. Obviously, the CPU plays a main part in this consumption. However, the RAM and disk operations are energy-consuming also, and can not be put aside. Besides, other parameters can modify the real energy consumed by a Cloud such as air conditioning management, network devices, ...

III. Using machine learning techniques for an accurate energy model

A. Approach overview

We realize, through these experiments and state of the art, that we cannot have total confidence in the simulations tools based on *spec.org* regarding the energy consumption of the whole system.

To overcome this lack of accuracy, we design a new energy model in which the parameter value can be refined using learning techniques. This energy model becomes a part of an adaptation engine controlled by an autonomic loop.

Step 1. Like the regular MAPE-K loop, we start by monitoring our managed element which is our Cloud infrastructure. Thanks to sensors, we retrieve precise metrics about what's going on in the datacenter (i.e. hosts usage, VM placement, VM workload, ...). All this monitoring information about the Cloud state are captured in a model.

Step 2. Then, we process all this information and the analyze module performs reasoning on the symptoms provided by the monitor. The analysis step determines the optional changes needed to bring the system in the ideal state (e.g. more energy-efficient, no SLA violations, high performance). Typically, in our solution, this step is achieved through metaheuristics reasoning on the underlying model. For instance, a genetic algorithm manipulates a Cloud configuration instanced as a model. On the basis of this model we apply variators such as mutation and crossover. This produces a new model which is then evaluated by the fitness function. The fitness functions are used to determine how good is the evaluated Cloud configuration. In our case we try to design a fitness function able to evaluate the energy consumption as our goal is to reduce this consumption. This fitness function predicts the Cloud consumption based on the potential Cloud configuration.

Step 3. Once the ideal state is determined, the planification examines the changes requested and structures the actions needed to achieve desired goals and objectives.

Step 4. Finally, the execute module performs the workflow of actions on the managed system using the dedicated effectors.

We slightly modify the loop, by adding a feedback loop for the knowledge (in our case the energy consumption simulator) which uses a Cloud configuration to determine the energy consumption. We introduce the use of machine learning techniques at runtime on the data collected by the monitor module. Continuously, the monitoring collects various metrics from the infrastructure and these data are forwarded to the knowledge module, so it can process it and refine the energy model using supervised learning algorithm. The knowledge keeps the analyze module updated with the latest energy model, so that analyze's algorithm can perform accurate evaluation on the runtime model. Here's, in figure 3, an illustration of the deployed MAPE-K loop:

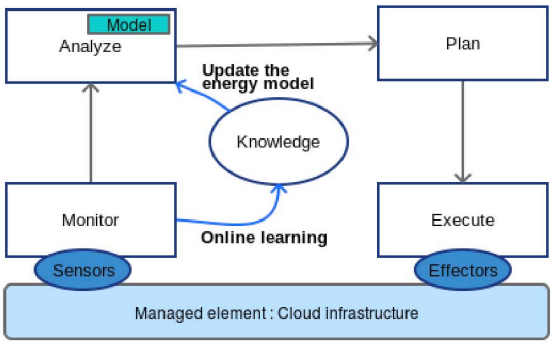

Figure 3. Customized autonomic loop

B. Cloud model

1) Cloud configuration model design: The model we use at runtime to represent the Cloud configuration and its characteristics aims at being as precise as possible. This model is mainly inspired by previous experiment based on Occi [1], CloudML [15] or Kevoree [14] to manage Cloud

configuration. In this experiment, we design a Cloud configuration meta-model that perfectly fits our need. However, we can conduct the same work using CloudML or Occi.

This model allows us to represent the mapping of the virtual machines on the servers, the SLA constraints and the different hosts load of the datacenter. Using this model we can reason on a Cloud configuration, perform mutations, crossovers and validity checks. We designed this model from scratch, representing a real Cloud infrastructure from the hardware layer to the application level. Indeed, a Cloud is one or more datacenters, with a network configuration and a set of applications to be run. Inside a datacenter, we have got multiple host nodes and routers. Host nodes are a composition of different hardware types (e.g. CPU, RAM, HDD, SSD) and network interfaces. Moreover, a host node has power management and consumption specifications according to its hardware. It hosts a collection of virtual machines, which have different flavors in terms of CPU, RAM or disk requested. Finally, VMs can run applications and therefore, consume the resources provided by host nodes.

2) Cloud configuration model implementation: To build an efficient data structure at runtime to reason on the Cloud configuration. We currently use KMF framework[2] [17]. This framework proposes a set of generators for an efficient implementation of models@runtime and provides mechanisms to store time series values when we have to store, in our case, a set of Cloud configuration that slightly varies.

C. Designing and refining the energy consumption model

As a result of the wide variety of server configurations, it is error prone to build a specific model matching all this heterogeneity. We rather opted for a self-learning model, needed to be trained for a given hardware, and then predicting with accuracy the energy values for this specific hardware.

To introduce the feedback loop, we use a set of sensors to measure the real energy consumption (PDU) and the set of agents to follow CPU, memory, disk usage per VM and per server. Basically, every compute node has a software agent installed next to the hypervisor. This software is a customized OpenStack Ceilometer compute agent [2]. It is a metrics collector, installed on each compute node of the datacenter. It polls metering data and instances statistics from the compute node through libvirt. Moreover, it monitors the host system, collecting various metrics on the node (e.g. CPU utilization, RAM usage, disk used). The compute agent forwards all the metrics to the central agent, responsible for aggregating them. Continuously, this software agent feeds our machine learning algorithm with the collected data.

These sensors allow us to associate an energy consumption value for a given physical server utilization (in terms of CPU utilization, RAM usage, disk I/O, network activity).

[2]http://kevoree.org/kmf/

Below, a detailed sequence of actions performed by every compute node agent:

- On the server, we monitor the current CPU utilization, RAM usage, volume of read and writes on the disk and volume of network data received and sent.
- With the PDU we get the corresponding energy consumed by the server
- Every second we retrieve the metrics from the server and the PDU
- Metrics collector stores the tuple (%cpu, %ram, read, writes, recv, sent, Watts)

Below, in figure 4, is depicted the metrics collection in the infrastructure.

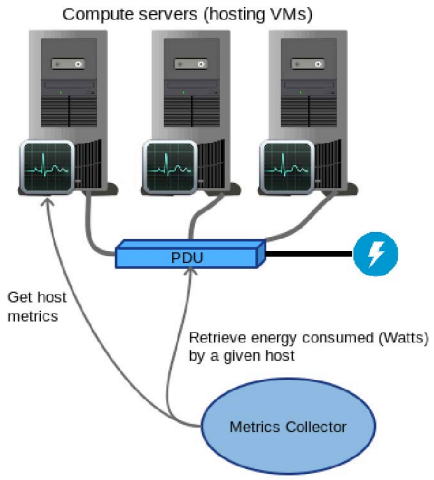

Figure 4. Host metrics collection

Based on this information, we use machine learning mechanisms to design a new energy model for the Cloud datacenters, to build accurate models in order to run realistic simulations.

In order to predict precise energy values during the Cloud optimization, we need to train our model beforehand, prior to the adaptation process, using the tuples associating the current host utilization to an energy consumption. Hence, it is necessary to let the agents gather a few thousands records prior to the start of adaptations. Note that, once this step is done, the algorithm keeps collecting records from the agents and thus can refine its energy cost model across time.

To train the model, we choose to use the Multivariate Adaptive Regression Spline [16]. First, we tried the basic multivariate linear regression and the multivariate polynomial regression function but it did not fit our dataset and the results were not relevant (we get up to 40 Watts of difference between prediction and real measurements). We decide to use the MARS algorithm as it looks promising in term of

performance time and RMSE[3] [9].

Multivariate Adaptive Regression Spline is a statistical technique popularized by Friedman for solving regression-type problems, with the main purpose to predict the values of a continuous dependent variable from a set of independent variables. It is a nonlinear regression modeling method, popular in the area of data mining because it does not assume any particular type or class of relationship (e.g., linear, logistic, etc.) between the predictor variables and the dependent variable of interest. Indeed, MARS can compute a model even in situations where the relationship between the predictors and the dependent variables is non-monotone and difficult to approximate with parametric models.

We submit to MARS the initial dataset of records, it computes the prediction equation and then continuously gather agents records to improve its predictions. Therefore, we are able to determine an accurate energy model for each type of hardware in the Cloud datacenter using the Multivariate Adaptive Regression Spline algorithm. The evaluation function, responsible for determining the energy consumption of a given Cloud configuration instanced as a model, can be formalized as follows:

$$\begin{aligned} E_{total} &= E_{servers} + E_{network} \\ &= \sum_{h=host}^{'} predict(h) + E_{network} \end{aligned}$$

Where:

- E_{total} denotes the total energy consumption of the Cloud infrastructure
- $E_{servers}$ refers to the energy consumption of physical servers/host nodes
- $E_{network}$ designates the energy consumption of network entities (i.e. routers, switches, ...)
- $predict(h)$ refers to the computed prediction function used to determine a server consumption from its metrics

The machine learning process does not take into account the datacenter network and its inherent energy consumption (due to the different equipment solicited). We decided to rely on the state of the art to evaluate the consumption of the several network entities. As mentioned in [20], the energy consumption of a switch is not proportional to the traffic load. Moreover, the network consumption depends on its topology. Therefore, we reuse existing network energy model coupled with our prediction function to determine the total energy consumed by the Cloud infrastructure

IV. Validation

A. Experimental protocol

In order to get a first energy model of the managed Cloud infrastructure, we need to gather several utilization thresholds of the different servers and their corresponding energy consumption. Note that we have to gather sparse data, representing different utilization levels of the server's hardware (i.e. CPU, RAM, disk, network) so that the prediction process can be trained for different cases. To this end, we stress the compute nodes of the Cloud infrastructure to mimic random and variable workloads. The software used to consume server resources is *stress-ng* version *0.03.15*. Using this tool, we generate some CPU utilization, RAM usage, disk operations and every compositions of them. Additionally, we produce some network activity using Bash scripts downloading and uploading several GNU/Linux distributions ISO images. At the same time we enable the software agents on the servers to collect the hardware utilization values and the corresponding energy consumption.

Here is an example of the training data gathered for a given host node:

CPU	RAM	reads	writes	recv	sent	Watts
15.5	2.3	2932	64	1	1	131
91.4	2.3	3188	14628	3473	44	189
...	...	...	...	...	...	...
6.2	1.5	259	0	2	1	101

Note that CPU and RAM are exprimed through a percentage, disk reads and writes, as well as network received and sent are exprimed in Kilobytes. Last, energy unit is the Watt.
Thanks to this data and the MARS algorithm, we obtain an energy model for a given server. Below are the results for the DELL PowerEdge R620 (with 3.13.0-24-generic x86_64 kernel), using the MARS open-source implementation [26], [5] (often referred to as Earth), with an initial dataset of 3000 records and a continuous learning for a week:

$$\begin{aligned} E_{host} = 170.965 &+ 0.794 * max(0, cpu - 11.900) \\ &- 2.625 * max(0, 11.900 - cpu) \\ &- 9.997 * max(0, 6.800 - ram) \\ &+ 0.009 * max(0, cpu - 11.900) * max(0, 42.000 - sent) \end{aligned}$$

Where:

- E_{host} is the total energy consumption of a given host
- cpu refers to the current host CPU utilization
- ram refers to the current host RAM usage
- $sent$ denotes the volume of network sent data (in Kb)

Given the equation predicting the Watts consumed by a loaded host, we can notice that it only depends on the CPU utilization, the RAM usage and the volume of sent data through the network. The MARS algorithm applied to our dataset determines that the disk and network features were not dominant. Thus, they are not relevant for predicting the energy consumed, despite having an effect on the system.

[3] Root-Mean-Square Error

We just need the dominant features which are CPU, RAM and network sent for the energy model computed by the MARS algorithm.

B. Energy model results

To validate our new energy model issued from the learning techniques, we evaluate it on a real dataset experiment. The initial dataset has been split in two parts. We keep 70% of the data to train our model through the MARS regression algorithm. The remaining 30% are meant to be compared in order to validate our energy model. For each set of input variables of the validation dataset, we compute the predicted consumption using our model, and we compare this value with the reality from the measured value.

The results look promising as we get an average error of 3,8% between the effectively measured values and the predicted ones which improve the accuracy comparing to CloudSim. This result permits to answer positively to $RQ2$ (see Introduction).

V. Threats to validity and discussions

We are quite surprised not to see the disk I/O as dominant features in the prediction equation computed by the MARS algorithm. Even though it is certain the disk operations are energy consuming, MARS evicts the input variables related to the disk. This can be explained by the fact that the volume of disk operations was quite constant. It can be due to our experimental protocol; perhaps pure sequential disk access is not realistic, and workloads must contain more random patterns, to stress magnetic disks as in real conditions.

Another point of interest is the good processing time of the MARS algorithm. It computes a large amount of data to determine the prediction equation in a fast time. In our last test, the training data is composed of one thousand records and MARS is able to produce the prediction equation in approximately 0.007 second.

Next step is to take into consideration the VM live migration energy overhead. Indeed, switching from one Cloud configuration to another introduces potential VM migrations, and these migrations consume additional energy on both the source and the destination node [19].

Last, now we have got a more accurate and realistic energy model, it can be embedded in the simulator or through models@runtime and it will evaluate simulated Cloud configurations with more precision. Nevertheless, we want to prove that a more accurate energy model can lead to different optimization results when used in a Cloud optimization problem For example, will a genetic algorithm evaluate with more precision the energy consumption of a Cloud configuration instantiated as a model using the proposed energy model? And thus, take better and smarter decisions in order to increase the energy efficiency of the Cloud infrastructure.

Note that all the source code about the MARS machine learning, the experiments and the validation are available at http://bcom.barais.fr.

VI. Conclusion and future works

Using energy simulator at runtime in a MAPE-K loop to self-optimize a Cloud infrastructure to maximize power efficiency and performance while maintaining predictable and reliable behavior, and at the same time responding appropriately to environmental and system changes such as hardware failures and varying workloads seems to be a promising idea. In this field, this paper presents the experimental results obtained to demonstrate that the use of *spec.org* values for the parameterization of the simulators does not create accurate results. It also presents an approach that uses machine learning techniques to refine Cloud simulation engine initial parameters and we show on an internal Cloud infrastructure that we can improve the Cloud simulation.

In the current work, we are doing experiment to quantify the energy consumption we save in optimizing the energy consumption simulator using machine learning techniques on an in-production Cloud infrastructure. In this current work, we are also working on the integration of this approach on top of iCanCloud. Next extension would be to take into account the VM migration cost and the network energy consumption cost in the energy cost model and autonomic loop. This approach is integrating to the OpenStack Watcher project[4] developed within the lab.

Acknowledgment

This work has been achieved within the Institute of Research & Technology b<>com, dedicated to digital technologies. It has been funded by the French government through the National Research Agency (ANR) Investment referenced ANR-A0-AIRT-07.

References

[1] Open Cloud Computing Interface - OCCI. *Online: http://occi-wg.org/*, 2011.

[2] *Ceilometer developer documentation*, 2015 (accessed May 11, 2015). http://docs.openstack.org/developer/ceilometer/.

[3] *Kevoree Project*, 2015 (accessed May 11, 2015). http://kevoree.org/.

[4] *The Network Simulator - ns-2*, 2015 (accessed May 11, 2015). http://www.isi.edu/nsnam/ns/.

[5] *Orange : Regression with Earth*, 2015 (accessed May 11, 2015). http://pythonhosted.org/orangecontrib.earth/orangecontrib.earth.html.

[4] https://wiki.openstack.org/wiki/Watcher

[6] *Spec.org - Dell PowerEdge R620*, 2015 (accessed May 11, 2015). http://www.spec.org/power_ssj2008/results/res2012q2/power_ssj2008-20120320-00444.txt.

[7] *stress-ng*, 2015 (accessed May 11, 2015). http://kernel.ubuntu.com/~cking/stress-ng/.

[8] *Stress project page*, 2015 (accessed May 11, 2015). http://people.seas.harvard.edu/~apw/stress/.

[9] Ajith Abraham and Dan Steinberg. Mars: Still an alien planet in soft computing? In *Computational Science-ICCS 2001*, pages 235–244. Springer, 2001.

[10] Arif Ahmed and Abadhan Saumya Sabyasachi. Cloud computing simulators: A detailed survey and future direction. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 866–872. IEEE, 2014.

[11] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50, January 2011.

[12] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, UKSIM '08, pages 126–131, Washington, DC, USA, 2008. IEEE Computer Society.

[13] FeiFei Chen, Jean-Guy Schneider, Yun Yang, John Grundy, and Qiang He. An energy consumption model and analysis tool for cloud computing environments. In *Proceedings of the First International Workshop on Green and Sustainable Software*, GREENS '12, pages 45–50, Piscataway, NJ, USA, 2012. IEEE Press.

[14] Donia El Kateb, François Fouquet, Grégory Nain, Jorge Augusto Meira, Michel Ackerman, and Yves Le Traon. Generic cloud platform multi-objective optimization leveraging models@run.time. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 343–350, New York, NY, USA, 2014. ACM.

[15] Nicolas Ferry, Alessandro Rossini, Franck Chauvel, Brice Morin, and Arnor Solberg. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In Lisa O'Conner, editor, *Proceedings of CLOUD 2013: 6th IEEE International Conference on Cloud Computing*, pages 887–894. IEEE Computer Society, 2013.

[16] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[17] Thomas Hartmann, Francois Fouquet, Gregory Nain, Brice Morin, Jacques Klein, Olivier Barais, and Yves Le Traon. A native versioning concept to support historized models at runtime. In *(MODELS'14) 17th International Conference on Model Driven Engineering Languages and Systems*, 2014.

[18] Dzmitry Kliazovich, Pascal Bouvry, and SameeUllah Khan. Greencloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.

[19] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster computing*, 16(2):249–264, 2013.

[20] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. A power benchmarking framework for network devices. In *NETWORKING 2009*, pages 795–808. Springer, 2009.

[21] Michael Maurer, Ivona Brandic, and Rizos Sakellariou. Self-adaptive and resource-efficient sla enactment for cloud computing infrastructures. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 368–375. IEEE, 2012.

[22] John C McCullough, Yuvraj Agarwal, Jaideep Chandrashekar, Sathyanarayan Kuppuswamy, Alex C Snoeren, and Rajesh K Gupta. Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf*, volume 20, 2011.

[23] Haque Monil, Mohammad Alaul, Romasa Qasim, and Rashedur M Rahman. Energy-aware vm consolidation approach using combination of heuristics and migration control. In *Digital Information Management (ICDIM), 2014 Ninth International Conference on*, pages 74–79. IEEE, 2014.

[24] A. Nuñez, J. L. Vázquez-Poletti, A. C. Caminero, J. Carretero, and I. M. Llorente. Design of a new cloud computing simulation platform. In *Proceedings of the 2011 International Conference on Computational Science and Its Applications - Volume Part III*, ICCSA'11, pages 582–593, Berlin, Heidelberg, 2011. Springer-Verlag.

[25] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, 46(4):47:1–47:31, March 2014.

[26] Jason Rudy. *GitHub : py-earth project*, 2015 (accessed May 11, 2015). https://github.com/jcrudy/py-earth.

[27] Sébastien Varrette, Mateusz Guzek, Valentin Plugaru, Xavier Besseron, and Pascal Bouvry. Hpc performance and energy-efficiency of xen, kvm and vmware hypervisors. In *Proceedings of the 2013 25th International Symposium on Computer Architecture and High Performance Computing*, SBAC-PAD '13, pages 89–96, Washington, DC, USA, 2013. IEEE Computer Society.