A Workload Prediction based Controller for Energy-Efficient Resource Allocation in Virtualized Data Centers

Adrian Cosinschi, Cristian Botau, Tudor Cioara, Ionut Anghel, Ioan Salomie
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{adrian.cosinschi, cristian.botau, tudor.cioara, ionut.anghel, ioan.salomie}@cs.utcluj.ro

***Abstract*-In this paper we approach the problem of data centers' high energy consumption by proposing a prediction based controller for optimal and energy efficient allocation of the hardware computing resources to the incoming workload. The controller analyzes the incoming workload virtual tasks' requests for hardware computing resources and classifies them in predefined tasks classes. For each class, the tasks' arrival rates are computed and the next period workload is predicted using a neural network. An optimal green data center situation is determined, in which all the predicted virtual tasks are accommodated and the hardware computing resources utilization is minimized. By comparing the current and predicted data center situations, the controller decides and constructs an adaptation action plan to accommodate the incoming workload in an energy efficient manner. The controller implementation is tested on a real test case data center and significant energy savings are reported.**

***Keywords-green data centers; energy efficiency; neural networks; workload prediction; workload consolidation*;**

I. INTRODUCTION AND RELATED WORK

The energy used by the data centers has drastically increased in recent years, being directly related with the number of hosted servers and their workload [1]. The data centers are starting to adopt virtualization and the cloud computing paradigm that allow to host multiple virtualized applications on the same physical server and to provision the required computing resource capacity on demand. In this new context, one state of the art solution to the problem of data center computing resources underutilization is the use of resource consolidation. Resource consolidation in data centers aims at combining the workloads that are executed on different machines so that an optimal number of computing resources are always used [2]. Using virtual machine migration, the workload can be consolidated on a smaller number of physical machines allowing for servers, or even entire operation nodes, to be completely shut down [3]. As shown in [4], the greatest challenge for consolidation techniques is deciding which workload should be combined on a common physical server since resource usage, performance, and energy consumption are not additive. To save energy in a virtualized data center using consolidation three main problems need to be addressed [3]: (1) VMs (Virtual Machines) placement - establishing the set of VMs executed by each server, (2) load balancing - determining the optimal request for computing resources at various servers and (3) capacity allocation - dynamically allocating server resources to VMs.

The state of the art consolidation methods, techniques and algorithms can be classified in three different types taking into account the consolidation time [4]: static consolidation, semi-static consolidation and dynamic consolidation. In the static consolidation the workload is dispatched and it is not further migrated [5]. This approach is used in case of static workload (i.e. doesn't change in time). In semi-static consolidation the server workload is consolidated once every day or every week [6]. The dynamic consolidation deals with very dynamic workloads (i.e. often change in time). It requires a run-time manager that should deploy and migrate workload applications, wake-up or turn-off servers as a response to workload variation [7]. Authors of [8] propose a hotspot detection algorithm that establishes when the VMs need to be migrated and their new location. A migration manager establishes the resources required by overloaded VMs and uses a greedy algorithm for determining the set of moves needed to migrate the VMs to under-loaded servers. In [9] a consolidation methodology that uses machine learning to deal with uncertain information is presented. Previous server behavior data is used to predict and estimate the current power consumption and to improve the scheduling and consolidation decisions. In [10], a technique for VM migration based on fuzzy logic is presented. The technique has two main steps: (1) the servers are sorted from the most over-loaded to the idle ones using a fuzzy decision making algorithm and (2) the VMs running on the over-loaded servers are migrated using a Multi Criteria Decision Making technique. An autonomic resources allocation technique for maximizing the revenue while balancing the cost of using the computing resources (including energy costs) is proposed in [11].

The problem of optimally allocating computational resources while considering different constraints like the energy efficiency or QoS levels is a NP-hard problem. Consolidation models based on the bin packing problem can be also considered [12]. Two different well known heuristics were proven to be useful for the bin packing based consolidation [13]: the best-fit decreasing and the first-fit decreasing. Cluster techniques are designed to obtain maximum SLA profits while minimizing the cost of using the resources including the energy consumed by the data center [14].

Regarding workload prediction, the existing techniques are usually based on time series analysis [15]. The prediction approach from [16] considers a workload time series to be comprised of a trend, a cyclical component (extracted using the workload's periodogram and autocorrelation) and some noise.

In this paper we approach the problem of data centers' high energy consumption by proposing a prediction based controller for green management of data center IT computing resources. The controller analyzes the incoming workload virtual tasks QoS requests for data center computing resources and classifies them in predefined tasks classes. For each class the tasks arrival rates are computed and the workload for next period is predicted using neural networks. The controller then determines an optimal data center situation in which all the predicted virtual tasks are accommodated on the data center so that the computing resources utilization is minimized. Comparing the current data center situation with the optimal determined one, the controller constructs an action plan that once executed will bring the data center in a green optimal state. Four different types of actions are used: virtual tasks migration/deployment and server turn-on/off.

II. Prediction based Management Controller Arhitecture

This section presents the conceptual architecture of the proposed management controller (see Fig. 1) for virtualized applications executed in a data center such that the SLA is guaranteed and the total energy consumption is reduced as much as possible. We consider that the data center workload is composed of virtualized tasks annotated with requests for data center computing resources. Virtualization is used as an abstraction level, because it allows us to manage the server's running tasks uniformly, without worrying about application dependencies and low-level details. The data center is seen as a set of physical servers on which the workload virtual tasks can be executed. Each data center server is described using the available amount of its computing resources (CPU, MEM).

The *Virtual Task Classifier* module analyzes each virtual task QoS requests (CPU, MEM) for data center computing resources and classifies them in a predefined class. The predefined classes of virtual tasks are stored in the *Virtual Tasks Classes Repository* database. A task class C_i is defined using two parameters: (1) $CPU_{range} = [CPU_{low}, CPU_{high}]$ representing the class possible processor values interval and (2) $MEM_{range} = [MEM_{low}, MEM_{high}]$ representing the class possible memory values interval. A task is classified as part of a predefined class if and only if its annotated requests for CPU and MEM computing resources belong to the defined class intervals. The *Workload Predictor* module uses the information about the workload tasks membership to a specific predefined class and predicts for each of them the arrival rate that will be experienced in the near future. Based on this prediction and on the current state of the data center, the *Virtual Tasks to Physical Server Mapper* module determines a new placement of the virtualized tasks already deployed on the physical servers and a possible placement of the tasks predicted to be deployed in the near future. When generating the virtual tasks placement, the *Virtual Tasks to Physical Server Mapper* takes into account the data center computing utilization aiming at lowering the energy consumption. The *Action Plan Creator* module compares the generated placement of the virtualized tasks for the next time period with the current *Data Center State* for constructing an action plan to achieve that placement. The *Virtual Tasks Manager* analyses the *Virtual Task Queue* workload and if it matches the predicted one it executes the action plan generated by the *Action Plan Creator* module. If no matching is found (miss prediction case), the *Virtual Tasks Manager* module will first fit deploy the virtual tasks on the available data center computing resources resulting in a higher energy payload. The *Workload Predictor* module accuracy influences the management controller ability to save energy.

III. Workload Prediction Method

In this section the proposed method for predicting the data center future workload is presented. The method is implemented in the *Workload Predictor* module. The developed prediction method uses the past data center workload stored in the *Workload Repository* database, and the current workload classified tasks to predict the arrival rate for each of the predefined task classes. The prediction process also takes into consideration some special events like the holidays (*Special Events Database*). The predicted task arrival rate for a class *C* represents the expected number of tasks classified as part of class *C* that should be accommodated on the data center in the next time period. The output of the *Workload Predictor* module is a vector $W_{predicted} = (Ar_{C_1}, Ar_{C_2} \dots Ar_{C_n})$ where Ar_{C_i} represents the predicted arrival rate of task class C_i for the next time period.

The prediction method is based on determining patterns of workload by modeling the task arrival rate as a function of its previous values and using historic workload data stored in the *Workload Repository* database. Due to the fact that the task arrival rate is highly correlated with time, the prediction method takes into consideration the workload time expressed as current hour, day of week and day of month.

A. Workload Repository and Special Events Database

The Workload Repository contains historic data center workload stored as past task arrival rate per task class. It stores the workload data using four levels of details with different granularity: (1) Five Minutes Level Of Detail (FML) stores 5 minutes average task arrival rate in the past 2 hours, (2) Half An Hour Level Of Detail (HHL) stores 30 minutes average task arrival rate in the past 24 hours, (3) Hour Level of Detail (HL) stores 1 hour average task arrival rate in the past month, (4) Day Level of Detail (DL) stores 1 day average task arrival rate in the past year. The Special Events Database contains information about special events. It is used by the Workload Predictor to learn a workload behavior that is correlated with holidays like Christmas, New Year's Eve or other events that may influence the data center workload.

B. Workload Predictor Internal Architecture

The *Workload Predictor* is composed of several *Master Predictor* components, one for each predefined task class, which, on their turn, are composed of four *Basic Predictor* components each having as inputs one of the four types of information contained in the *Workload Repository*.

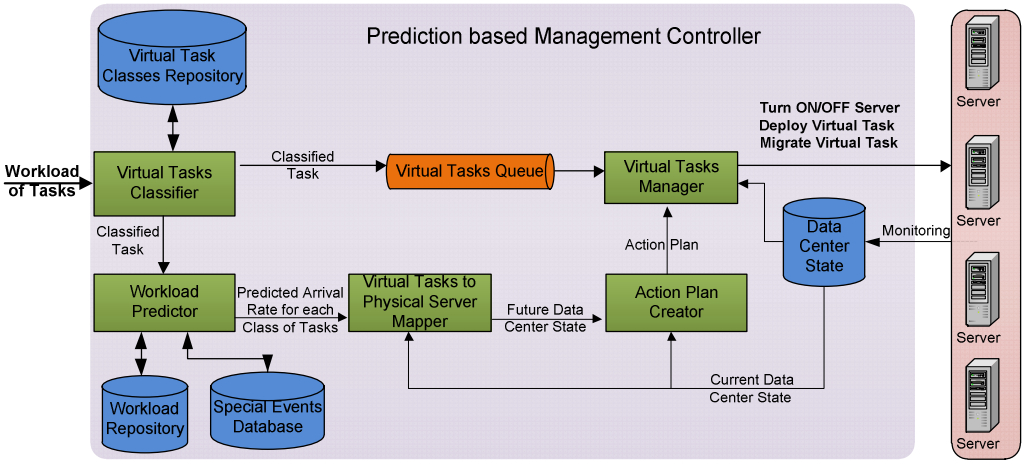

Figure 1. Prediction based controller architecture

Fig. 2 presents the design of the *Master Predictor* associated to a task class C_i. The *Master Predictor* inputs are: the C_i task class arrival rate stored in the *Workload Repository,* the current system time, and the special events stored in the *Special Events Database.* The output of the *Master Predictor* is the task arrival rate for the associated class in the next time period. This value is a weighted sum of the predicted values provided by each of the four *Basic Predictor* modules. The weight assigned to each *Basic Predictor* is proportional to the accuracy of its past predictions.

The *Basic Predictor* analyzes the workload variations, learns the workload behavior and predicts the future workload values. More generally, the *Basic Predictor* learns a function $W_{predicted} = f(t, W_{history}, SE)$, where t is the current time (such as current time of day, current day of week, current month), $W_{history}$ is the historical workload information and *SE* represents the set of inputs providing information about special events. The *Basic Predictor* implementation is done using a feed forward multilayer artificial neural network having one hidden layer. The advantage of this type of neural network is that it can express any continuous function. The training algorithm used for each *Basic Predictor* is back propagation algorithm [17] where the learning rate is fixed and was determined empirically by testing. Each perceptron of the artificial neural network uses the sigmoid transfer function shown in relation 1.

$$f(x) = \frac{1}{1 - e^{-x}} \tag{1}$$

The input values fed into the neural network are normalized to the interval [0, 1]. The number of inner nodes for each *Basic Predictor* is fixed and configurable. During the simulations we found that a value of 0.7 for the learning rate works well. Also we have determined that the optimal number of inner nodes of the neural network should be 1.5-2 times the number of inputs. For a higher number of inner nodes we risk over fitting, while for a lower number of nodes the learning performance drops. For the implementation of the artificial neural network and of the learning algorithm we used the *Neuroph* framework [18].

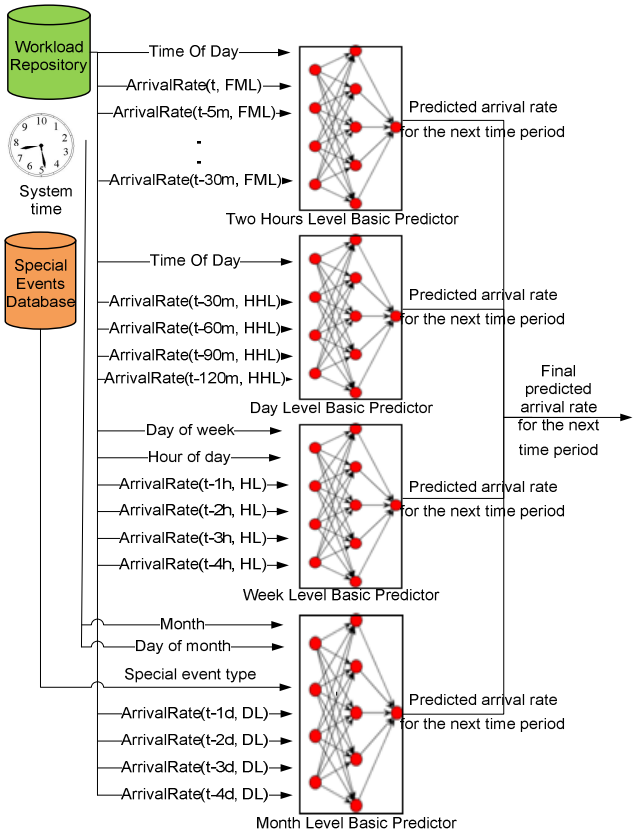

Figure 2. Master predictor design

The training set for each *Basic Predictor* is chosen based on three training parameters: *sampling interval size*, *training element size* and *training set window size*. The *sampling interval size* is the time interval during which the arrival rate is computed by counting the number of tasks that need to be accommodated in the data center in that interval. The *training element* is a set of consecutive arrival rates that form a workload pattern and is used in the artificial neural network training process. If the *training element* contains *N* workload tasks then the first *N-1* arrival rates are used as inputs to the artificial neural network and the *Nth* arrival rate is used as

output of the artificial neural network. The *training set window* is a time interval during which the training elements are sampled. The *training set* is obtained by sweeping the *training set window* and sampling a training element each *sampling interval*. Based on the *training window size*, we defined the following *Basic Predictors*: *Hour Level, Day Level, Week Level* and *Month Level*.

IV. Data Center Green Management Algorithm

In this section the algorithm (see Fig. 3) used by the management controller to improve the data center computing resources utilization and as a consequence its energy consumption is detailed. The algorithm is implemented by three main modules of the management controller: *Virtual Tasks to Physical Server Mapper, Action Plan Creator and Virtual Tasks Manager*.

Algortihm: Data Center Green Management

input: workloadTasks - the tasks waiting to be deployed
T - current time period
SC^T - current data center state in T
begin
1. **if** (workloadTasks!=NULL) **then**
2. newTasks = getNextTasks (workloadTasks)
3. newTasksClass = classifyTasks (newTasks)
4. notify (workloadPredictionEvent, WorkloadPredictor)
5. en-queue (newTasks, TasksQueue)
6. **endif**
7. **if** (workloadPredictionEvent) **then**
8. predictedTasks = Prediction (newTasks, newTasksClass)
9. SC^{T+1} = genFutureSCState (SC^T, predictedTasks)
10. actionPlan = genActionPlan (SC^T, SC^{T+1})
11. Notify (actionPlanEvent, VirtualTaskManager)
12. **endif**
13. **if** (actionPlanEvent) **then**
14. currentTasks = de-queue(TasksQueue)
15. **if** (currentTasks == predictedTasks) **then** execute actionPlan
16. **else** firstFitDeploy **(**currentTasks**)**
17. **endif**
end

Figure 3. The data center management algorithm

The *Virtual Tasks to Physical Server Mapper* module uses as input the predicted workload values for the next time period (algorithm lines 1-6) and the *Current Data Center State* to determine an energy efficient placement of the virtualized tasks already deployed on the physical servers and a possible placement of the tasks predicted to be deployed in the near future (see lines 7-10). In other words it determines a *Future Data Center State* in which all the predicted virtual tasks are deployed so that the computing resources are optimally used and the tasks QoS requests are fulfilled. To evaluate the quality of the *Future Data Center State* in terms of energy consumption and deployed tasks QoS requests fulfillment, a metric called *Future Data Center State Utility* is defined and used. The *Action Plan Creator* module compares placement of the virtualized tasks for the next time period described by generated *Future Data Center State*, with the current placement described by the *Current Data Center State*. It constructs an action plan that by execution will place the incoming and already deployed virtual tasks according to the *Future Data Center State* description (see lines 10 and 11). The action plan contains four types of actions: deploy virtual task, migrate virtual task, turn-on server, turn-off server. The action plan is then passed to the *Virtual Tasks Manager* module which deals with its execution (see lines 13-17). If the current tasks received by the service center are not the ones predicted then the first fit deploy is used as fallback strategy (see lines 15-16).

A. Data Center State Representation

The Data Center State in a certain time period *T* is described by the following tuple:

$$SC^T = (X^T, Alloc^T(VT)) \quad (2)$$

where X^T is a bi-dimensional array which elements x_{ij} describe the virtualized tasks deployment on the data center servers (if $x_{ij} = 1$, the task VT_i is placed on the server S_j, otherwise $x_{ij} = 0$) and $Alloc^T(VT)$ is a one-dimensional array which elements represent the resources allocated to each virtualized task VT_i. We will denote by M the total number of servers in the data center and by N^T the total number of virtualized tasks executed by the data center servers in T time period. Also, we define a one-dimensional array Y^T (that can be computed from X^T) describing the state of each server in time period T (if $y_j = 1$, the server S_j is up and running and executes one or more virtualized tasks, otherwise $y_j = 0$).

A data center server S_j can accommodate one or more virtualized tasks as long as there are sufficient server computational resources $R(S_j)$ so that all the deployed tasks QoS requests are fulfilled. A data center is in a *consistent state* if and only if the following four conditions are fulfilled.

Condition 1. All the data center workload virtualized tasks VT_i, need to be hosted on the data center servers (relation 3).

$$\sum_{j=1}^{M} x_{ij} = 1 \quad \forall\, i \in [1, N] \quad (3)$$

Condition 2. The amount of computational resources allocated to a certain virtualized task ($Aloc(VT_i)$) is higher or equal to the virtualized task QoS requests for servers computational resources $Req_{QoS}\,(VT_i)$ (relation 4).

$$Aloc(VT_i) \geq Req_{QoS}\,(VT_i) \quad \forall\, i \in [1, N] \quad (4)$$

Condition 3. The computational resources allocated for the virtualized tasks hosted by a data center server must not exceed the server total amount of computational resources (relation 5).

$$\sum_{i=1}^{N} (x_{ij} * Aloc(VT_i)) \leq R\,(S_j) \quad \forall\, j \in [1, M] \quad (5)$$

A data center is in an *energy efficient green state* if and only it is in a consistent state and the following relation holds:

$$\sum_{j=1}^{M} y_j * R\left(S_j\right) - \sum_{i=1}^{N} Aloc(VT_i)) \rightarrow 0 \quad (6)$$

B. Data Center State Utility Metric

The *Data Center State Utility* metric $U[SC^T \rightarrow SC^{T+1}]$ evaluates a *Future Data Center State* (SC^{T+1}) generated by *Virtual Tasks to Physical Server Mapper* module compared with the *Current Data Center State* (SC^T) in terms of energy efficiency and deployed tasks QoS requests for server computing resources fulfillment.

The metric has two main components (relation 7): (1) a QoS related component $QoS[SC^{T+1}]$ representing the virtualized tasks degree of QoS request satisfaction for future data center state, and (2) an energy related component $E[SC^T \rightarrow SC^{T+1}]$ representing the future data center state utility obtained by deploying the predicted workload tasks, consolidating the current data center state virtualized tasks and reducing the number of servers up and running.

$$U[SC^T \rightarrow SC^{T+1}] = QoS[SC^{T+1}] + E[SC^T \rightarrow SC^{T+1}] \quad (7)$$

The QoS utility component is computed as the sum of the degree of QoS request fulfillment of each virtualized task executed in the data center, as shown in relation 8.

$$QoS[SC^{T+1}] = \sum_{i=1}^{N} f\left(Aloc(VT_i), Req_{QoS}\left(VT_i\right)\right) \quad (8)$$

The energy utility component $E[SC^T \rightarrow SC^{T+1}]$ reflects the costs of: (1) maintaining a physical server up and running in SC^{T+1}, (2) turning-on a new server in SC^{T+1} that was turned-off in SC^T, (3) deploying the new predicted virtualized tasks and (4) migrating old virtualized tasks.

C. Determining the Future Data Center State

In this section, we present the technique that is used by the *Virtual Tasks to Physical Server Mapper* to determine an optimal (maximum utility) *Future Data Center State* SC^{T+1} in which all the data center virtualized tasks (predicted/already deployed) are placed on the data center servers so that the computing resources are optimally used and the tasks QoS are fulfilled.

The total number of tasks that have to be accommodated by the data center in the next period N^{T+1} will be the sum between the number of virtualized tasks already deployed that didn't finish their execution and the total number of tasks predicted to arrive in the next period. For each predicted virtual task the allocated computing resources ($Alloc^{T+1}(VT_i)$) will correspond to the higher limit of the task class to which it has been classified by the *Virtual Tasks Classifier*. This will assure that the tasks QoS request for computing resources are always fulfilled thus the $QoS[SC^{T+1}]$ part of the *Future Data Center State Utility* metric is maximized. To determine the placement of virtualized tasks on data center servers X^{T+1} in SC^{T+1} the *Virtual Tasks to Physical Server Mapper* follows the rules presented in Fig. 4. This will assure that the next data center state will be a green state and in consequence the $E[SC^T \rightarrow SC^{T+1}]$ part of the *Future Data Center State Utility* metric is maximized.

> ***Rule 1***. *Minimize the number of up and running servers that accommodate virtual tasks.*
> ***Rule 2***. *Use firstly for the placement of new virtual tasks the servers that are already running and only if necessary turn-on other turned-off servers.*
> ***Rule 3***. *Place the virtual tasks whenever possible on the same servers as in the past. Only when their resource requirements cannot be satisfied on the old locations or when it is possible to turn-off a turned-on server perform live migration actions.*

Figure 4. Rules to determine X^{T+1} in the Future Data Center State

The problem of computing X^{T+1} such that the rules stated above are satisfied under the constraint that the next data center state is consistent will be reduced to the well-known *Single Source Capacitated Facility Location Problem (SSCFLP)*. The *SSCFLP* requests to determine a number of facilities that must be opened in order to provide a useful service to a set of customers, to choose the site locations of these facilities from a set of possible locations, and to establish which customers will be serviced by which facility such that a global cost is minimized. This global cost comprises the cost of opening the facilities and the cost of satisfying the demands from customers. Also the facilities must have enough capacity to service all the customers assigned to them. In our case, the facilities are equivalent to the data center servers while the customers are equivalent to the workload virtual tasks executed by the data center. The capacity of a facility is mapped to the amount of computing resources of a server. The customer's demands are similar to the virtual tasks QoS requests for servers computing resources. To open/close a facility at a certain site location is translated to the action of turning-on/off a server. The virtual task deployment and migration actions are equivalent to the customer-facility mapping. The coefficients from the global cost which needs to be maximized by the *SSCFLP* problem, are computed in such a manner that the maximization of the global cost will result in the minimization of the energy consumption cost. The *SSCFLP* is a NP-hard problem, but efficient algorithms were proposed to compute a solution close to the optimal one. For our implementation, we used a genetic algorithm described in [19].

V. Case Study

To evaluate the energy saving capabilities of the proposed controller in a real data center environment, a test case data center with the following IT computing resources was used: (1) a server cluster composed from five physical servers on which virtualized tasks are executed and (2) and a power meter (IPM 3005) which measures the power consumed by the five servers.

On each server we have installed CentOS 5.5 as operating system. On top of the operating system resides the KVM hypervisor which deals with the virtualization of the server hardware resources. The hypervisor also offers functionalities

for monitoring and managing a virtual machine. On top of the KVM hypervisor we have used OpenNebula, a middleware for managing virtualized clouds. OpenNebula offers functionalities for managing all the virtual machines deployed on the data center servers, such as virtual machines deployment on a specific server, virtual machines migration, etc.

The OpenNebula middleware facilities are used by the management controller module *Virtual Tasks Manager* which executes the action plan decided by the *Action Plan Creator Module*. The *Virtual Task Classifier* module considers 16 different tasks classes that represent all the combinations of CPU and memory ranges presented in Table I.

TABLE I. THE PROCESSOR/MEMORY RANGES

Range / Resource	Very Low	Low	High	Very High
CPU (No. Cores)	[1, 2]	[3, 4]	[5, 6]	[7, 8]
MEM (MB)	(0,1536]	(1536, 2930]	(2930, 4608]	(4608,6144]

In our case study, for three of the task classes a workload with a repetitive pattern regarding the arrival rate (having a repetition period of 20 minutes) has been generated. The arrival rates for these classes are presented in Table II.

TABLE II. ARRIVAL RATES PER TASK CLASS

Class / Time	Very Low CPU & Memory	Low CPU & Memory	High CPU & Memory
0-5 min	2	2	1
5-10 min	0	1	0
10-15 min	1	0	0
15-20 min	0	0	0

Fig. 5 shows the power consumption values measured while executing the generated workload for about 25 minutes, in two situations: (1) the data center is managed by our proposed algorithm (marked with red) and (2) the data center is managed by OpenNebula first fit algorithm (marked with blue).

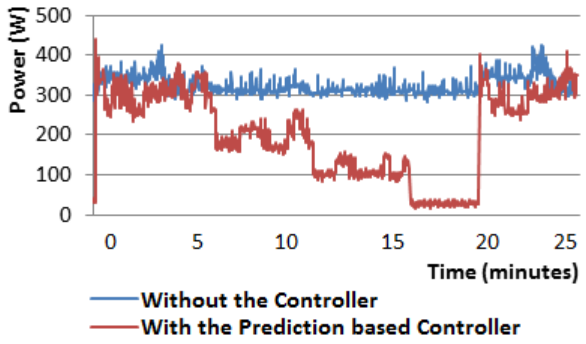

Figure 5. Data center instant power consumption variation

The results show that when the data center is managed by our proposed algorithm, energy saving of about 59.76% is obtained during the periods of time when the workload is predicted correctly. Even though this percentage is really high it should be noted that the data center IT facility aspect was neglected during simulation. In a data center, the IT facilities usually consume about 60% from the data center total power consumption. Thus our solution's power saving represents about 24% from the total power consumption of a data center.

VI. CONCLUSIONS

In this paper we approach the problem of data centers' high energy consumption by proposing a prediction based controller for optimal allocation of the hardware computing resources to the incoming workload and consequently optimizing energy consumption. The tests show that the data center energy saving improvement when the controller was used is about 24% from the total energy consumption of the data center.

ACKNOWLEDGMENT

This work has been partially supported by the GAMES project [20] and has been partly funded by the European Commission IST activity of the 7th Framework Program (contract number ICT-248514).

REFERENCES

[1] U.S. Environmental Protection Agency, "EPA Report on Server and Data Center Energy Efficiency", ENERGY STAR, 2007.
[2] N. E. Jerger, D. Vantrease, M. Lipasti, "An Evaluation of Server Consolidation Workloads for Multi-Core Designs", Proc. of IEEE Int. Symposium on Workload Characterization, 2007.
[3] S. Srikantaiah, A. Kansal and F. Zhao, "Energy Aware Consolidation for Cloud Computing", Microsoft Research, 2009.
[4] A. Verma, G. Dasgupta, T. Kumar Nayak, et al., "Server workload analysis for power minimization using consolidation", USENIX Annual technical conference, 2009.
[5] Q. Zhu, J. Zhu et al., "Power-Aware Consolidation of Scientific Workflows in Virtualized Environments", High Performance Computing, Networking, Storage and Analysis, ISBN: 978-1-4244-7557-5, 2010.
[6] M. Uddin and A. A. Rahman, "Server Consolidation: An Approach to Make Data Centers Energy Efficient & Green", International Journal of Scientific & Engineering Research, Volume 1, Issue 1, 2010.
[7] N. Bobroff, A. Kochut, et al., "Dynamic placement of virtual machines for managing SLA violations", Integrated Network Management, 2007.
[8] T. Wood, P. Shenoy, A. Venkataramani, M. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration", Computer Networks Journal Special Issue on Virtualized Data Centers, 2009.
[9] J. Berral, I. Goiri, R. Nou, et al., "Towards energy-aware scheduling in data centers using machine learning", Int'l Conf. on Energy-Efficient Computing and Networking, 2010.
[10] M. Tarighi, S.A. Motamedi, et al., "A new model for virtual machine migration in virtualized cluster server based on Fuzzy Decision", 11th Int. Conference on Advanced Communication Technology, 2009.
[11] D. Ardagna, M. Trubian, L. Zhang, "SLA based resource allocation policies in autonomic environments", Journal of Parallel and Distributed Computing, volume 67, 2007.
[12] R. Anderson, E. Mayr, M. Warmuth, "Parallel Approximation Algorithms for Bin Packing", Stanford University, 1988.
[13] Y. Ajiro, and A. Tanaka, "Improving Packing Algorithms for Server Consolidation", Proc. of the Computer Measurement Group's, 2007.
[14] G. Pacici, M. Spreitzer, A. N. Tantawi, A. Youssef, "Performance Management for Cluster-Based Web Services", IEEE Journal on Selected Areas in Communications, December 2005.
[15] G. Jenkins, G. Reinsel, G. Box, "Time Series Analysis: Forecasting and Control", Prentice Hall, 1994.
[16] D. Gmach, J. Rolia, L. Cherkasova, A. Kemper, "Workload Analysis and Demand Prediction of Enterprise Data Center Applications", Proc. of IEEE Int. Symposium on Workload Characterization, 2007.
[17] R. Rojas, "A graph labelling proof of the backpropagation algorithm", Communications of the ACM, Vol. 39, Dec. 1996.
[18] Neuroph Framework, http://neuroph.sourceforge.net/.
[19] R. Mauricio, J. P. de Sousa, A. Viana, "Metaheuristics: computer decision-making", Kluwer Academic Publishers, 2010.
[20] GAMES FP7 Research Project, http://www.green-datacenters.eu/.