# A Green Scheduling Policy for Cloud Computing

JORDI VILAPLANA, FRANCESC SOLSONA, JORDI MATEO, IVAN TEIXIDÓ AND JOSEP RIUS,
Dept. of Computer Science & INSPIRES, University of Lleida
FRANCESC ABELLA, Santa Maria Hospital & IRB Lleida

This paper presents a power-aware scheduling policy algorithm called Green Preserving SLA (GPSLA) for cloud computing systems with high workload variability. GPSLA aims to guarantee the SLA (Service-Level Agreement) by minimizing the system response time and, at the same time, tries to reduce the energy consumption. We present a formal solution, based on linear programming, to assign the system load to the most powerful Virtual Machines, while respecting the SLA and lowering the power consumption as far as possible. GPSLA is thought for one node load-aware and jobs formed by embarrassingly parallel heterogeneous tasks.

The results obtained by implementing the model with the IBM CPLEX prove the applicability of our proposal for guaranteeing SLA and saving energy. This also encourages its applicability in High Performance Computing due to its good behavior when scaling the model and the workload. The results are also highly encouraging for further research into this model in real federated clouds or cloud simulation environments, while adding more complexity.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Evaluation / methodology*; H.1.2 [**Models and Principles**]: User/Machine Systems—*Human Information Processing*; I.5.1 [**Pattern Recognition**]: Models—*Neural Nets*

Additional Key Words and Phrases: Green cloud computing, SLA, power-aware scheduling, linear programming

**ACM Reference Format:**

Jordi Vilaplana, Francesc Solsona, Jordi Mateo, Ivan Teixidó, Francesc Abella and Josep Rius. 2014. A Green Scheduling Policy for Cloud Computing. *ACM Trans. Appl. Percept.* 2, 3, Article 1 (May 2010), 9 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

This work is supported by the MEyC under contract TIN2011-28689-C02-02. The authors are members of the research group 2009-SGR145, funded by the Generalitat de Catalunya.

Author's address: J. Vilaplana, Dept. of Computer Science & INSPIRES, University of Lleida, Jaume II 69, 25001 Lleida, Spain; email: jordi@diei.udl.cat; F. Solsona, Dept. of Computer Science & INSPIRES, University of Lleida, Jaume II 69, 25001 Lleida, Spain; email: francesc@diei.udl.cat; J. Mateo, Dept. of Computer Science & INSPIRES, University of Lleida, Jaume II 69, 25001 Lleida, Spain; email: jmateo@diei.udl.cat; I. Teixidó, Dept. of Computer Science & INSPIRES, University of Lleida, Jaume II 69, 25001 Lleida, Spain; email: iteixido@diei.udl.cat; F. Abella, Santa Maria Hospital & IRB Lleida, Avda Rovira Roure, 44, 25199 Lleida, Spain; email: abella@gss.scs.es ; J. Rius, Dept. of Computer Science & INSPIRES, University of Lleida, Jaume II 69, 25001 Lleida, Spain; email: josep.rius@udl.cat.


DOI: http://dx.doi.org/10.1145/0000000.0000000

## 1. INTRODUCTION

In cloud computing, hardware and software services can be added and released dynamically in order to guarantee an SLA (Service-Level Agreement) to clients [1]. An SLA is an agreement between a service provider and a consumer where the provider agrees to deliver a service to the consumer under specific terms, such as time or performance. In order to comply with the SLA, the service provider must monitor the QoS (Quality of Service) closely through such performance metrics as response or waiting time, throughput or makespan [1].

We focus our attention on the response time as the QoS metric. In this scenario, the SLA contract usually states that the consumer only pays for the resources and services used according to the agreed QoS requirements at a given price [2]. Studying and determining SLA-related issues is a big challenge, mainly due to the complex nature of cloud computing and especially its high variability [4].

Our proposal, Green Preserving SLA (GPSLA), is designed to lower power consumption [5] as much as possible. At the same time, GPSLA is aimed at guaranteeing a negotiated SLA and power-aware [3] solutions, leaving aside such other cloud-computing issues as variability [4], system security [5] and availability [6]. Job response time is perhaps the most important QoS metric in a cloud-computing context [2]. For this reason, it is also the QoS parameter chosen in this work. In addition, despite good solutions having been presented by some researchers in the literature dealing with QoS [7; 8] and power consumption [9; 10; 11; 12], the model presented aims to obtain the best scheduling taking both criteria into account.

There is a great deal of work in the literature on linear programming (LP) solutions and algorithms applied to scheduling, as the one presented in [17; 18]. Other remarkable work was performed in [13], where authors designed a Green Scheduling Algorithm that integrated a neural network predictor in order to optimize server power consumption in Cloud Computing. Moreover, in [14] authors developed a power-aware virtual machine scheduling mechanism that targeted both load-balancing and temperature-balancing. In [15] authors discussed a two-level task scheduling mechanism based on load balancing in cloud computing. Also, in [16] authors proposed a genetic algorithm that takes into account both makespan and energy consumption. Our main objective is the designing of an LP scheduling algorithm to minimize power consumption and maximizing SLA guaranties (based on the response time as the QoS performance metric) at the same time.

An important contribution of this paper is the way we model the power of the virtual machines in function of its workload. We rely on the work done in [19], where the authors formulate the problem of assignment of persons from various groups to different jobs who may complete them in minimum time as an stochastic programming problem. The job completion times were assumed to follow a Gamma distribution. To model the influence of the workload we weighted the powerful of the Virtual Machine by a load factor determined by an Erlang distribution (equivalent to a Gamma). Finally, we obtained an stochastic programming problem and transformed to an equivalent deterministic problem with linear objective function.

The remainder of the paper is organized as follows. In the Green Preserving SLA Schedulers section we present our main contributions, a sort of scheduling policies. These proposals are arranged in order of increasing complexity. The experimentation showing the good behavior of our cloud model is presented in the Results section. Our proposal was tested with the *CPLEX* mathematical optimizer, which provides tools to implement the mathematical models presented in the Green Preserving SLA Schedulers section. Finally, the Conclusions section outlines the main conclusions and possible research lines to explore in the near future.

## 2. GREEN PRESERVING SLA SCHEDULERS

Our scheduling proposals are based on linear programming (LP). LP consists of trying to find an objective function (OF) representing one, or if possible more than one, performance criteria. Multiple performance criteria can be taken into account in order to choose the optimal scheduler. The most widely used are the minimization of the power consumption of the cloud and the mean response time of tasks. These are also the ones chosen in this article.

LP applied to scheduling deals with finding one assignment that maximizes or minimizes the OF. This will give the assignment of tasks to VMs or the consolidation of VMs to nodes that maximizes the gains in the chosen performance metric. The formulated solution, the OF and the constraints, can be resolved by any solver, such as IBM CPLEX, or the open-source *lp_solve*.

Cloud scheduling can be split into two more specific phases, the scheduling itself and the consolidation phase. The scheduling phase is the one that assigns tasks to virtual machines (VMs). As an example of this, simple requests/tasks entering a cloud system will be scheduled for execution in one of the VMs that make it up. Then the consolidation phase ensures the efficient use of resources avoiding under-utilized VMs. However, energy savings are related to these two phases, thus encouraging the use of one-step scheduling.

A two-step scheduler can give very distinct results to the ones given by a one-step scheduler. For example, a two-step task scheduler can assign tasks to 2 VMs residing on two different nodes. However, a single-stage task scheduler can only assign these to one VM, thus enabling the idle node to be stopped, and so saving power. The cause of this failure is to consider the return time in the first step and power saving in the second. Incorporating energy savings is the main reason behind that design decision.

We next present our scheduling proposals, called Green Preserving SLA (GPSLA). To better understand the most complete proposal better, we present 3 different scheduler models, arranged according to their increasing complexity.

### 2.1 GPSLA. One node

This policy assigns as many requests (tasks) as possible to the most powerful Virtual Machines (VMs), leaving aside the remaining ones. The unused VMs could be then turned off to save energy. The method is based on the computing capacity of the VMs, making assignments in descendent order to their computing power, processing as many requests as possible per unit of time, thus prioritizing the preservation of the SLA contract (for some QoS) with the clients.

We take a cloud made up of a single node, allocating $V$ heterogeneous VMs. Each VM $VM_v$ has a specific amount of Memory $M_v$. This restricts the workload each VM can host. Given $T$ tasks, all of which are supposed to be homogeneous and, for reasons of simplicity, have the same computing and memory requirements, a single generic unit.

We define the *relative computing power* ($\Delta_v$) of a $VM_v$ as the *normalized score* of such a VM. Formally, given $V$ VMs, $\Delta_v = \frac{\delta_v}{\sum_{k=1}^{V} \delta_k}$, where $\sum_{k=1}^{V} \Delta_v = 1$. $\delta_v$ is the score (i.e. the computing power) of VMv. Although $\delta_v$ is a theoretical concept, there are many valid benchmarks to obtain it (i.e. Linpack, Lapack or SPEC). Linpack (available in C, Fortran and Java) for example, is used to obtain the number of floating-point operations per second. Note that the closer the relative computing power is to one (in other words, more powerful), the more likely it is that the requests will be mapped into such a VM.

The scheduling is obtained by considering the maximum computing power ($t_v \Delta_v$) of each $VM_v$. So, we firstly assign tasks to the most powerful VMs. This is a design decision in order to optimize the cloud resources, based on optimizing some QoS performance metrics, such as response or waiting time, throughput or makespan.

As mentioned above, the chosen QoS metric is the response time. Furthermore, by doing it this way, priority is also given to guaranteeing SLA until a certain limit is reached (the one imposed by the cloud capacity) is also prioritized. This objective can be formally defined with the following linear programming model:

$$max(\sum_{v=1}^{V} t_v \Delta_v) \tag{1}$$

$$s.t. : \sum_{v=1}^{V} = T \tag{2}$$

$$t_v \leq M_v, \forall v \leq V \tag{3}$$

Equation 1 is the *Objective Function* (OF) to be maximized. Equality in Equation 2 and inequality in Equation 3 are the constraints of the objective function variables. Given the constants $T$ (the total number of requests or tasks), and $\Delta_v$ and $M_v$ for each $VM_v$, the solution that maximizes ***OF*** will obtain the values of the variables $t_v$, representing the number of tasks assigned to $VM_v$. Thus, the $t_v$ obtained will be the assignment found by this model.

### 2.2 GPSLA. One node load-aware

Going a step further in the previous model, the loss of power in function of the VM workload is also taken into account. We first consider that VM efficiency rises with the load until some a certain number number of tasks is reached. From then on, the efficiency starts falling asymptotically towards zero. We can model this behaviour with an *Erlang* distribution. Erlang is a continuous probability distribution with two parameters $\alpha$ and $\lambda$. The parameter $\alpha$ is called the shape parameter, and the parameter $\lambda$ is called the rate parameter. These parameters depend on the VM characteristics. When the parameter $\alpha$ equals 1, the distribution simplifies to the exponential distribution. The *Erlang* probability density function is:

$$E(x; \alpha, \lambda) = \lambda e^{-\lambda x} \frac{(\lambda x)^{\alpha-1}}{(\alpha-1)!} \forall x, \lambda \geq 0 \tag{4}$$

We consider that the Erlang modelling parameters of each VM can easily be obtained empirically in an easy way (i.e. by increasing the workload and measuring the mean response times). Figure 1 shows various Erlang plots for some $\alpha$ and $\lambda$ values by varying the $x$, designed in this case to represent the workload.

The Erlang distribution was developed to examine the number of telephone calls which that might be made at the same time to the operators on a switchboard. We use it here to weigh the computing power ($t_v \Delta_v$) of each $VM_v$ by multiplying it by an Erlang distribution. Here, the $x$ Erlang parameter is replaced by $t_v$ in order to model this distribution in function of each VM workload. Thus giving the new model:

$$max(\sum_{v=1}^{V} t_v \Delta_v E(t_v; \alpha, \lambda)) \tag{5}$$

$$s.t. : \sum_{v=1}^{V} = T \tag{6}$$

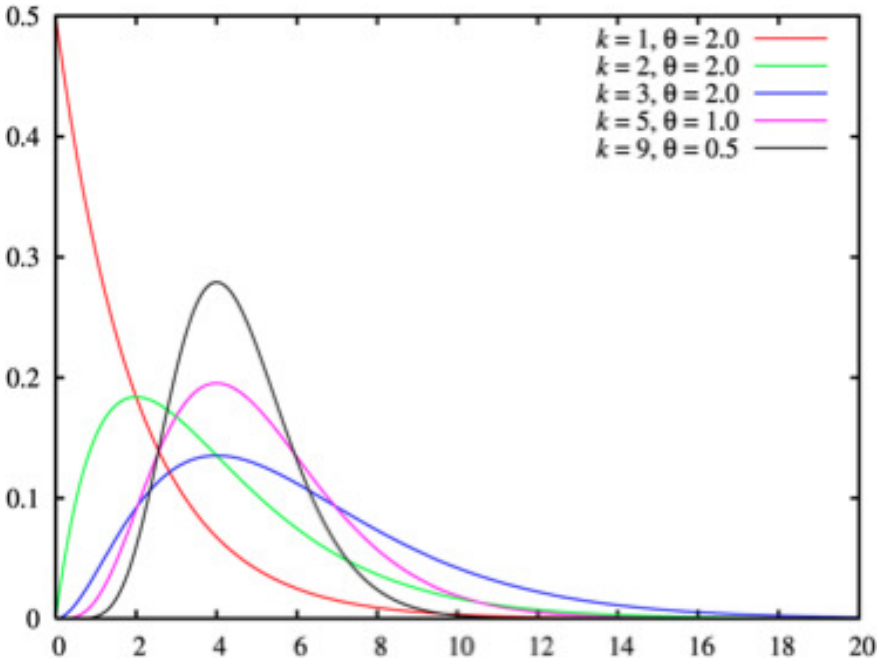


Fig. 1. Erlang plots for different $\alpha$ and $\lambda$ values.

$$t_v \leq M_v, \forall v \leq V \tag{7}$$

### 2.3 GPSLA. One node load-aware and heterogeneous tasks

Going even further, we are now also interested in to considering in addition the heterogeneity of task computing requirements. In other words, each task $t_i$ has its *Processing cost* $P_{vi}$, representing the execution time of task $t_i$, in $VM_v$ with respect to the execution time of task $t_i$ in the less powerful $VM_v$ (this is, with the lowest $\Delta_v$). $M_{vi}$ is defined as the amount of Memory allocated to task $t_i$ in $VM_v$. We can suppose that Memory requirements do not change between VMs, so $M_{vi} = M_{v'i} \forall v, v' \leq V$. We define the Boolean variable $t_{vi}$, representing the assignment of task $t_i$ to $VM_v$. Then, the new linear model is:

$$max(\sum_{v=1}^{V}(\sum_{i=1}^{T} P_{vi}t_{vi})\Delta_v E(\sum_{j=1}^{T} P_{vj}t_{vj}; \alpha, \lambda)) \tag{8}$$

$$s.t. : \sum_{i=1}^{T} = M_{vi} \leq M_v \forall v \leq V \tag{9}$$

$$\sum_{v=1}^{V} t_{vi} = 1 \forall i \leq T \tag{10}$$

$P_{vi}t_{vi}$ takes into account the computing cost of task $t_i$ in its assigned $VM_v$. As $t_v$, representing the number of tasks assigned to $VM_v$ has been changed by the Booleans $t_{vi}$, which represent the

assignment of each task $t_i$ to $VM_v$, the Erlang function must also be changed in order to compute all the tasks assigned to each VM, so $t_v$ is changed to $\sum_{j=1}^{T} P_{vj} t_{vj}$.

## 3. RESULTS

Several experiments were performed in order to test the *GPSLA* results in different configurations and for the different stages of complexity of the scheduling policy (Sections 2.1, 2.2 and 2.3). The experimental results were obtained using the CPLEX mathematical optimizer.

The experimentation case shown here was defined with 3 VMs and 50 tasks. The VM configurations are shown in Table I.

Table I. VM configurations

| VM | Relative Computing Power ($\Delta v$) | Memory | Erlang Distribution |
|---|---|---|---|
| 1 | 0.45 | 50 | $\alpha = 3, \lambda = 8$ |
| 2 | 0.35 | 40 | $\alpha = 2, \lambda = 8$ |
| 3 | 0.2 | 10 | $\alpha = 2, \lambda = 4$ |

This simulation was performed by applying the GPSLA policy to the three models described.

### 3.1 GPSLA. One node

For the first model (GPSLA. One node), described in Section 2.1, the resulting assignation is shown in Table II.

Table II. Task distribution obtained with the "GPSLA. One node" model.

| VM 1 | VM 2 | VM 3 |
|---|---|---|
| 50 | 0 | 0 |

As the efficiency drop caused by overloading is not taken into account, all tasks were assigned to VM 1. As VM 2 and VM 3 are not used, they could be turned off to save energy.

In this case, the results obtained are consistent with the model, which tries to consolidate all the tasks in the most powerful virtual machine.

### 3.2 GPSLA. One node load-aware

For the second model (GPSLA. One node load-aware), described in Section 2.2, the Erlang distributions described in Table 1 was applied. The three Erlang charts can be seen in Figure 2, Figure 3 and Figure 4.

These Erlang distributions model the behavior of the VM with different amounts of load. To obtain the best task allocation considering the saturation of virtual machines in the addition of more and more tasks, it is compulsory the Erlang discretization. To realize this discretization, each number or different range of tasks must correspond to a specific discrete value. The set of all these values will be used to calculate the time of the optimization objective function. Thus the parameter relating to the Erlang always have a discrete value. This way, the stochastic programming problem is transformed to an equivalent deterministic problem with linear objective function.

It can be seen how, according to its Erlang distribution function, VM 1 reaches its maximum performance between 15 and 25 tasks. VM 2 and VM 3 will have a slightly different behavior and will reach their maximum with fewer tasks. Note that these distributions should be fine-tuned according to each specific system when working with real environments.

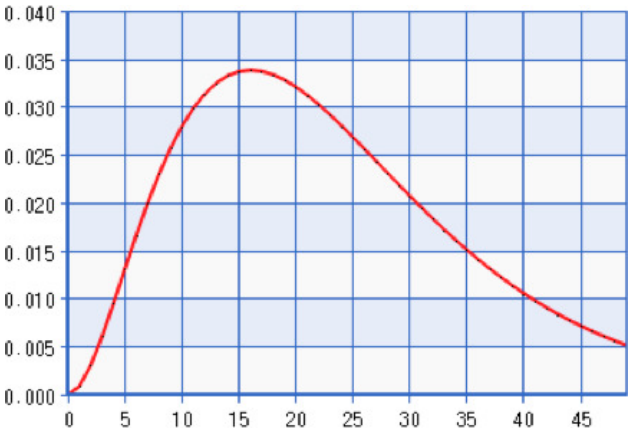


Fig. 2. Erlang distribution chart for VM 1.

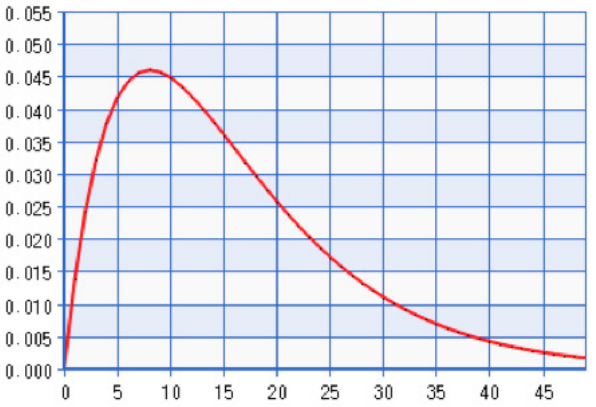


Fig. 3. Erlang distribution chart for VM 2.

Table III. Task distribution obtained with the ”GPSLA. One node load-aware” model.

| VM 1 | VM 2 | VM 3 |
|---|---|---|
| 25 | 17 | 8 |

The resulting assignation is shown in Table III. It can be seen that when taking into account the system overload the distribution of tasks among virtual machines changes. The scheduler now tries to put all the possible tasks in the most powerful VM, but when the performance starts decreasing significantly, it starts sending further tasks to the other available virtual machines. This behavior is more accurate as, although energy saving is a priority, we still want to preserve a certain degree of quality of service.

### 3.3 GPSLA. One node load-aware and heterogeneous tasks

The third model (GPSLA. One node load-aware and heterogeneous tasks), described in Section 2.3, shows the same behavior when tasks are defined with different weights. So the results proved consistent.

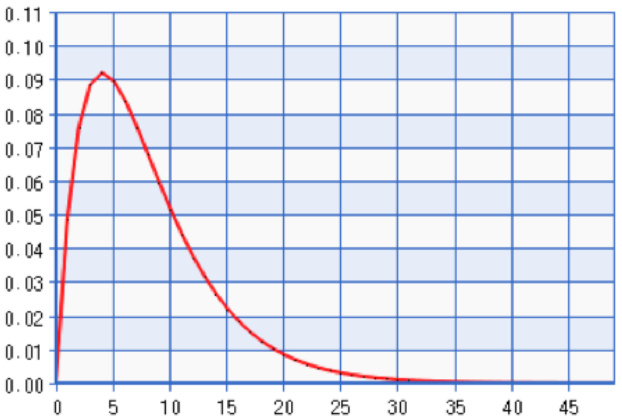


Fig. 4. Erlang distribution chart for VM 3.

## 4. CONCLUSIONS

This paper presents a cloud-based system scheduling mechanism able to respond successfully to a high degree of variability complying with low power consumption and SLA agreements. The complexity of the model developed was increased, thus adding more factors to be taken into account. The model was also tested using the CPLEX mathematical optimizer, and the results obtained proved consistent over a range of scenarios. However, these scenarios have to be tested in such real and simulated cloud environments such as OpenStack and CloudSim, and it is also needed to scale the physical and virtual resources of the system. So, the applicability of the algorithm for designing variability- and power-aware cloud systems was proven. To summarize, although more experiments are needed, these preliminary results corroborate the usefulness of our proposal. Future trends are in the direction of exploring different heterogeneous workloads (i.e. jobs/tasks with different computing needs and communication paradigms, and not only the embarrassingly parallel) by expanding the linear programming proposal. The upper and lower SLA boundaries should also be considered. Finally, we want to assess the design of cloud architectures (bearing in mind cloud federation) and VM consolidation. Nevertheless, we believe that this preliminary work is very encouraging and could be further developed and applied in real cloud platforms.

## ACKNOWLEDGMENTS

This work was supported by the MEyC under contract TIN2011-28689-C02-02. The authors are members of the research group 2009-SGR145, funded by the Generalitat of Catalunya.

### REFERENCES

M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. *A view of cloud computing. Commun. ACM, vol. 53, no. 4, pp. 50-58. 2010.*

A. Keller and H. Ludwig. *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services, J. Netw. Syst. Manage. 11(1) 57-81, 2003.*

R. Aversa, B. Di Martino, M. Rak, S. Venticinque, U. Villano. *Performance Prediction for HPC on Clouds. Cloud Computing: Principles and Paradigms. 2011.*

J. Varia. *Architection for the Cloud: Best Practices. Amazon Web Services. 2014.*

A. Iosup, N. Yigitbasi, D. Epema. *On the Performance Variability of Production Cloud Services. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'2011), pp. 104-113. 2011.*

K. V. Vishwanath, N. Nagappan. *Characterizing cloud computing hardware reliability. In Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10), pp. 193-204. 2010.*

M. Martinello, M. Kaâniche, K. Kanoun. *Web service availability: Impact of error recovery and traffic model. Journal of Reliability Engineering and System Safety, Vol. 89, n. 1, pp. 6-16. 2005.*

J. Vilaplana, F. Solsona, F Abella, R. Filgueira, J. Rius. *The cloud paradigm applied to e-Health. Bmc Medical Informatics and Decision Making. 2013.*

J. Vilaplana, F. Solsona, I Teixidó, F. Abella, J. Rius. *A queuing theory model for cloud computing. The Journal of Supercomputing. 2014.*

A. Beloglazov, R. Buyya. *Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers. Concurrency and Computation: Practice and Experience, vol. 24 pp. 1397-1420. 2012.*

D. Kliazovich, P. Bouvry, S. Khan. *GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing. 2010.*

J. Kaur, S. Vashist, R. Singh, G. Singh. *Minimization of Energy Consumption Based on Various Techniques in Green Cloud Computing. International Journal of Innovative Research in Computer and Communication Engineering. Vol. 2, Issue 3. 2014.*

K. Le, J. Zhang, J. Meng, R. Bianchini, Y. Jaluria, Thu D. Nguyen. *Reducing Electricity Cost Through Virtual Machine Placement in High Performance Computing Clouds. Proceeding SC '11 Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. Article No. 22. 2011.*

Truong Vinh Truong Duy, Y. Sato, Y. Inoguchi. *Performance evaluation of a Green Scheduling Algorithm for energy savings in Cloud computing. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW) 2010, pp. 1–8, 2010.*

Y. Mhedheb, F. Jrad, J. Tao, J. Zhao, J. Kołodziej, A. Streit. *Load and Thermal-Aware VM Scheduling on the Cloud. Algorithms and Architectures for Parallel Processing. Lecture Notes in Computer Science Volume 8285, 2013, pp 101-114. 2013.*

Y. Fang, F. Wang, J. Ge. *A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing. Web Information Systems and Mining Lecture Notes in Computer Science Volume 6318, 2010, pp 271-277. 2010.*

M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.-G. Talbi, A.Y. Zomaya, D. Tuyttens. *A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing, Volume 71, Issue 11, November 2011, pp. 1497–1508, 2011.*

J.Ll. Lérida, F. Solsona, P. Hernández, F. Giné, M. Hanzich, Josep Conde. *State-based predictions with self-correction on Enterprise Desktop Grid environments. Journal of Parallel and Distributed Processing, vol 71, n. 11, pp. 777–789, 2012.*

A. Goldman, Y. Ngoko. *A MILP Approach to Schedule Parallel Independent Tasks. International Symposium on Parallel and Distributed Computing, ISPDC '08, pp. 115–122, 2008.*

M.F. Khan, Z. Anwar, Q.S. Ahmad. *Assignment of Personnels when Job Completion time follows Gamma distribution using Stochastic Programming Technique. International Journal of Scientific & Engineering Research, Volume 3, Issue 3, 2012.*