Cooperative Scheduling Anti-load balancing Algorithm for Cloud : CSAAC

Cheikhou Thiam
IRIT
Universit Paul SABATIER
Toulouse, France F-31062
Email: Cheikhou.Thiam@irit.fr

Georges Da Costa
IRIT
Universit Paul SABATIER
Toulouse, France F-31062
Email: Georges.Da-Costa@irit.fr

Jean-Marc Pierson
IRIT
Universit Paul SABATIER
Toulouse, France F-31062
Email: Jean-Marc.Pierson@irit.fr

***Abstract*—In the past decade, more and more attention focuses on job scheduling strategies in a variety of scenarios. Due to the characteristics of clouds, meta-scheduling turns out to be an important scheduling pattern because it is responsible for orchestrating resources managed by independent local schedulers and bridges the gap between participating nodes. Likewise, to overcome issues such as bottleneck, overloading, under loading and impractical unique administrative management, which are normally led by conventional centralized or hierarchical schemes, the distributed scheduling scheme is emerging as a promising approach because of its capability with regards to scalability and flexibility. In this paper, we introduce a decentralized dynamic scheduling approach entitled Cooperative scheduling Anti-load balancing Algorithm for cloud (CSAAC). To validate CSAAC we used a simulator which extends the MaGateSim simulator and provides better support to energy aware scheduling algorithms. CSAAC goal is to achieve optimized scheduling performance and energy gain over the scope of overall cloud, instead of individual participating nodes. The extensive experimental evaluation with a real workload dataset shows that, when compared to the centralized scheduling scheme with BestFit as the meta-scheduling policy, the use of CSAAC can lead to a 30%61% energy gain, and a 20%30% shorter average job execution time in a decentralized scheduling manner without requiring detailed real-time processing information from participating nodes.**

Keywords: Energy, Heuristic, Virtual Machines, Cloud, Migration.

I. Introduction

Cloud computing delivers infrastructure, platform, and software (applications) as services that are made available to consumers in a pay-as-you-go model. In industry these services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) respectively. Many computing service providers including Google, Microsoft, Yahoo, and IBM are rapidly deploying data centers in various locations around the world to deliver Cloud computing services. Cloud service providers ensure that they can be flexible in their service delivery to meet various consumer requirements, while keeping the consumers isolated from the underlying infrastructure. Until recently, high performance has been the sole concern in data center deployments, and this demand has been fulfilled without paying much attention to energy consumption. However, an average data center consumes as much energy as 25,000 households [13]. Green Cloud computing is envisioned to achieve not only the efficient processing and utilization of a computing infrastructure, but also to minimize energy consumption [14]. To address this problem of minimizing energy and drive Green Cloud computing, data center resources need to be managed in an energy-efficient manner. Furthermore the increasing demand of computation resources has led to new types of cooperative distributed systems for cloud computing.

Given the increasing data center scales, such systems are faced with challenges in terms of scalability, autonomy, and energy-efficiency. In large-scale Cloud, the centralized approach is clearly unfeasible. Firstly, centralized scheduling [11] requires accurate, centralized information about the state of the whole system. Secondly, sites forming the grid maintain some level of autonomy, yet classic algorithms implicitly assume a complete control over individual resources. Thus, many of the existing attempts to design and implement cloud systems are still based on centralized architectures, have limited autonomy, and lack of energy saving mechanisms.

In contrast, decentralized scheduler [12] negates the limitations of centralized structures with respect to fault-tolerance, scalability, autonomy, and most importantly the adequacy for the Cloud computing environment. Jobs are submitted locally, but they can be migrated to another cluster, if the local cluster is overloaded. The possibilities of migration are, however, limited, so that migrated jobs do not overload the host system. A decentralized scheduling approach assumes that each entity is autonomous and has its own control that derives its scheduling decision based on its policies. However, if the decisions are taken by several independent units, it might be the case that these units aim at optimizing their own objectives rather than the performance of the system as a whole. Such situations call for models and techniques that take the strategic behavior of individual units into account, and simultaneously keep an eye on the global performance of the system. In most cases, self-interested entities have to cooperate to achieve their respective objectives, but any cooperation must be self-enforcing and not enforced by binding agreements through third parties.

Such a cooperative scheduler can be seen as a system including a set of consumers, e.g. applications or tasks, and a set of resources as energy and processing power. To make the most of all available resources, a proper distribution of tasks among the devices such as to optimize the energy consumption, represents a key issue to be addressed. The design and implementation of such task allocation (or task scheduling) strategy is an objective of this paper. The main design principle of our scheduler is to find a task allocation in order to maximize energy saving. Thus we propose an energy-aware task allocation scheme that distributes energy consumption among clusters by balancing the energy load among them. To this aim, we designed and implemented a two phase heuristic-based algorithm. This algorithm first tries to assign a task locally to the cluster that generated the execution request by minimizing energy consumed. If the task cannot be assigned locally, the second phase of the algorithm is performed by assigning the task to the most suitable node from the complete network of resources, minimizing the energy requirements of the system. We characterize the energy consumption of the system defining an energy efficiency model in which the energy costs of both computation and communication activities are taken into account.

There are several works in literature whose goal is to evenly distribute workload. But when the goal is to reduce energy consumption, this type of algorithms can lead to resources being largely underloaded, resulting in unnecessary energy consumption. This weakness is a major problem of load balancing algorithms.

A part of this work focuses on the extension of the MaGateSim simulator [7] as well as better support to energy aware scheduling algorithms. MaGateSim is based on GridSim toolkit [10] and Alea simulator [19], and abstracts the features and behaviors of complex fundamental grid elements, such as grid jobs, grid resources, and grid users. the simulator itself can be easily extended, and adopted for evaluating newly developed decentralized scheduling algorithms, models, or workflows. Community Aware Scheduling Algorithm (CASA) [10] makes job allocation decisions based on contacted nodes loads and real-time responses. In this case, each participating node needs to expose its resource utilization, during cooperation with other nodes. Furthermore, it is able to reschedule jobs through time, in order to adapt to the unpredictable performance changes of independent underlying resources. Cooperative scheduling Anti-load balancing Algorithm for cloud (CSAAC) extends CASA. CSAAC extends CASA adding migration of VMs and the ability to avoid underload and overload nodes. We have evaluated our network of schedulers using a simulation of the system Grid5000. To save energy, CSAAC provides a holistic energy-efficient task placement algorithm. Particularly, it integrates migration of jobs and mechanism which automatically switch off nodes, transitions them into a power-saving state (e.g. suspend), and wakes them up once required. The detection of underload and overload nodes is performed. To remedy this, consolidation is used. The experimental results have proven this algorithm to be energy-efficient.

The remainder of the paper is organized as follows. Section II discusses related work. Section II presents the Model and objectives. Section IV presents the Decentralized dynamic scheduling anti-load balancing Algorithm for grids (CSAAC). Section V presents the experimental results. Section VI Finally concludes the paper.

II. Related work

Generally meta-scheduling solutions are classified into three categories, namely the centralized, hierarchy, and decentralized schemes. In a centralized scheduling architecture [11], scheduling decisions are made by a central controller for all VMs. The scheduler maintains all information about the VM and keeps track of all available resources in the system. Centralized scheduling organization is simple to implement and easy to deploy. A. Beloglazov and Rajkumar Buyya [16] have proposed and evaluated heuristics for dynamic reallocation of VMs to minimize energy consumption, while providing reliable QoS. Their results show that the technique of dynamic reallocation of VMs and switching off the idle servers brings substantial energy savings and is applicable to real-world Cloud data centers. This work has not investigated setting the utilization thresholds dynamically according to a current set of VMs allocated to a host, leveraging multi-core CPU architectures, and decentralization of the optimization algorithms to improve scalability and fault tolerance. In order to avoid scheduling self competition, some Clouds only allows one scheduler to manage each virtual organization. However, Centralized scheduling organization is not adequate for the Cloud because of the nature of the Cloud computing environment. Again, these centralized services limit their scalability.

In distributed scheduling, there is a central manager and multiple lower-level entities. This central manager is responsible for handling the complete execution of a VM and assigning the individual VM to the low-level providers. Each lower-level entity scheduler is responsible for mapping the individual tasks into Cloud resources. R. Ranjan et al [4] [5] proposed a meta-scheduling framework. Each resource consumer may value various resources differently depending on its QoS based utility functions and may want to negotiate a particular price for using a resource based on demand, availability and its budget. An SLA is the agreement negotiated between a meta-scheduler, entitled the Grid Federation Agent (GFA), and the Local Resource Management System (LRMS) of the local sites in terms of acceptable job QoS constraints, such as job response time and budget spent. It highlights a bid-based SLA contract negotiation model. Furthermore, the contract net protocol [6] based SLA bids are restricted with a certain expiration time, and a variety of economic parameters such as setting price, user budget and deadline. A greedy backfilling heuristic is also proposed for application on the participating LRMSs during their cooperation with the meta-schedulers. The failure of the central manager results in entire system failure.

The decentralized meta-scheduling scheme allows each node to own a meta-scheduler which receive job submissions originated by local users, and to assign such jobs to the local resource management system, i.e., local scheduler. Meta-schedulers of different nodes are capable of exchanging information and sharing jobs between each other in order to balance the resource load amongst participating nodes. Besides the issue of efficiency and overhead, the decentralized scheme brings better scalability, compared to other scheduling schemes. C. Comito et al [8] proposed a task allocation scheme for mobile networks focusing on energy efficiency. To conservatively consume energy and maximize network lifetime they have introduced a heuristic algorithm that balances the energy load among all the devices in the network. Authors have implemented a prototype of the system and evaluated the scheduling strategy through simulation experiments. Results show that the proposed scheduler greatly enhances the performance of the system compared to time-based traditional schedulers like the round-robin. They achieved improvements in terms of network lifetime, number of active devices and number of completed tasks. Authors refer to a cooperative Energy-Aware Scheduling strategy that assigns computational tasks over a network of mobile devices optimizing the energy usage. We use the same type of decentralized architecture, but their objective is to find a task allocation that prolongs the network lifetime while our main objective is to minimize the energy consumption in Cloud.MaGateSim [7], a Simulation Environment for a Decentralized Grid Scheduler, is designed to be a decentralized grid scheduler that emphasizes on grid scheduler interoperation, and is complemented by a dynamic resource discovery approach on decentralized network. In order to share the jobs submitted from a local MaGate to other MaGates within the same grid community, a set of community scheduling relevant parameters are evaluated and discussed to address various job delegation scenarios between different MaGates. The same authors propose a decentralized dynamic scheduling approach named the community-aware scheduling algorithm (CASA) [10]. However, the problem has not been explored in the context of the optimization of energy consumption. In an attempt to remedy this issue, our work is based on CASA with a main purpose to optimize the energy gain and ensuring a good quality of service.

III. Model and objectives

The overall objective of the energy management policy is reducing energy consumption, while satisfying the users performance demand within Cloud. In this section, we discuss the Model, hypotheses, constraints and objectives.

A. Model and hypothesis

We consider an cloud environment compared of several clusters. Each application will run in a dedicated VM. Each VM has requirements in terme of computing power. For example in figure 1 job 1 neads 30% of it's host cpu. Hosts can be switched off to save energy.

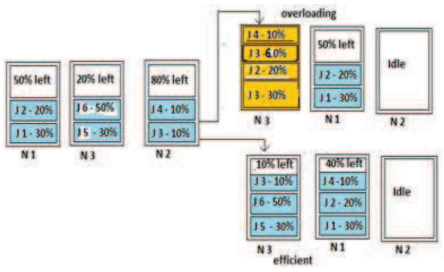

Fig. 1. Different contexts for a migration.

- **hypotheses**
  - Communications within a cluster and between other clusters for migration are considered as negligible
  - For each cluster there is at least one host and one VM.
  - Migration cost [21] is considered as the same in CloudSim. To migrate a VM, only RAM has to be copied to another node. The migration time depends on the size of RAM and the available network bandwidth. VM migration delay = RAM / bandwidth + C (C = 10 sec). Bandwidth is considered as constant.

We will use the classical linear model of power consumption in function of load :

$$\forall i, j \qquad P_{i,j} = P_{min}^{i,j} + c_{i,j}(P_{max}^{i,j} - P_{min}^{i,j})$$

Therefore the total power consumption of the system is:

$$P = \sum_{i=1}^{N} \sum_{j=1}^{H_i} P_{i,j}$$

To obtain energy consumed during a time slice, instantaneous power has to be multiplied by time. Total energy is then obtained by summing all the energy of those time slices.

B. Objectives

The main objective of our approach is to improve cloud's total energy efficiency by controlling cloud applications' overall energy consumption while ensuring cloud applications service level agreement. Therefore our work aims to satisfy several objectives :

- Ease of task Management : we design a system which is flexible enough to allow for dynamic addition and removal of servers. As system components can fail at any time, it is desirable for a system to heal in the event of failures without human intervention. Consequently, we aim at designing a system using self-healing mechanisms to enable high availability.
  - Conservation of the execution context : It must be possible to stop the execution process of the task and restart it where it has stopped. Execution time can be reduced when the job migrates to a more powerful node.
  - Slowdown prevention : Job slowdown increases the execution time and therefore increases the energy consumed by hosts and impact users.
- Energy Efficiency: One of our goals is to propose task placements management algorithms which are capable of creating idle times, transitioning idle servers in a power saving state and waking them up once required (e.g. when load increases). To measure the saturation of a cluster, we use the saturation threshold ε.

IV. Cooperative scheduling Anti-load balancing Algorithm for cloud (CSAAC)

A. Our threshold based Anti-load balancing model

Scheduling a workflow is a process of finding the mapping of tasks to the suitable resources so that the execution can be completed with the satisfaction of objective functions, such as execution time minimization. Existing workflow scheduling approaches are non-coordinated, where workflow schedulers perform scheduling related activities independent of the other schedulers in the system. They directly submit their tasks to the underlying Cloud resources without taking into account the current load, priorities, and utilization. This leads to over-utilization or a bottleneck on some valuable resources, while leaving others largely under-utilized. Further, brokering approaches do not have a coordination mechanism. This worsens the load sharing and utilization problems of Cloud resources. Cooperative decision making for scheduling in an open environment enables an optimized workflow execution considering the dynamic resource behavior in the Cloud.

The proposed algorithm in this article works by associating two threshold values with each host. When a host is under-loaded (load $<$ globally_defined_threshold), all its tasks are migrated to a comparatively more loaded host. we also use the over-loaded threshold ε, which we call saturation, to measure the saturation of a cluster,.

In dynamic load unbalancing schemes, the two most important policies are *selection policy and location policy*. Selection policy concerns the choice of the host to unload. Location policy chose the destination host of these moved tasks. An important characteristic of selection policy is to prevent the destination host to become overloaded. Also, migration costs must be compensated by the performance improvement.

B. The Community-Aware Scheduling algorithm (CASA)

As a multi-phase decentralized scheduling solution, CASA is comprised of the job submission phase responsible for job dissemination, as well as the dynamic scheduling phase responsible for iterative scheduling improving. Job submission phase is the first phase of the community-aware scheduling algorithm. Each time when a node i, receives a job k submitted by its local user, node i behaves as a requester node and uses algorithm $h_{request}$ to generate a request message $req_{i,k}$ for job k. Job characteristic information including estimated execution time ln_k and requested amount of Processor Elements (PEs) pe_k will be appended to the generated request message. Afterwards, request message message $req_{i,k}$ replicated and disseminated to each of the discovered remote nodes asking for the job delegation possibilities. For all nodes receiving the job delegation request message $req_{i,k}$, including the requester node i itself, are considered as responder nodes. Each responder node j needs to launch an algorithm named h_{accept} to decide whether node j is able and willing to execute the received job k. Algorithm $h_{request}$ takes various factors, such as the responder nodes capabilities and administrative preferences, into consideration to decide whether job k can be executed upon the responder node. If yes, each candidate, considered as responder node, computes an estimated completion time according to its current scheduling and resource status and delivers the information by means of an ACCEPT message. In addition, the estimated response time if job k is executed by node j is also appended to the generated ACCEPT message, which can be utilized by the requester node for responder node evaluation and selection. Each time a request message $req_{i,k}$ is generated by the requester node and disseminated to contactable remote nodes, the requester node waits and collects all received ACCEPT messages and invokes an algorithm h_{assign} to select a proper remote node to which to delegate the job. An assign will be generated and send to the assignee node, wherein the job k and its relative data are enclosed.

Furthermore, CASA is a collection of heuristic sub-algorithms, which are used to facilitate job scheduling across decentralized distributed nodes. For an arbitrary node i, due to the effect of the job submission phase, it has received a set of jobs from either local users submissions or remote nodes delegations. A rescheduling process helps CASA to adapt to the changes of both underlying resources and arriving jobs. Each time a node attempts to (re)assign a job to another node (or the same node) for execution according of its load, the assignment initiator is called the requester node, and the node receiving such a request is called the responder node.

C. Algorithm description

As a decentralized scheduling solution, the cooperate scheduling anti-load balancing algorithm in Cloud (CSAAC) is based on CASA [10]. CASA proposed algorithm adopts the promised job response time as the only criterion to evaluate the nodes capabilities. Participating nodes only need to calculate estimated response time for a concerned job and bid for the job delegation using the calculated and promised job response time. CSAAC add another criterion to evaluate the nodes capabilities, the node load.

Each responder node computes an estimated completion time according to its current scheduling and resource status, calculates the necessary energy, and delivers the information by means of an ACCEPT message. In addition, the node load is also appended to the generated message, which can be utilized by the requester node for responder node evaluation and selection.

The selected node is the best candidate node based on several parameters, such as the promised time to complete, energy consumed, the node load between under-load threshold and over-load threshold, node weight due to historical interaction records, etc. Furthermore, during the execution of the tasks, all the time the system verifies if there are under loaded or overloaded nodes.

CSAAC provides task scheduling strategy, which dynamically migrate tasks among computing hosts, transferring tasks from underloaded hosts to loaded but not overloaded hosts. It balances load of computing hosts as far as possible in order to reduce program running time.

The decision making algorithm behaves globally as follows:

- If total Vm load on the host j of cluster $i > \varepsilon$,host is over-loaded
- If total Vm load on the host j of cluster $i < \gamma$, host is under-loaded

Selection policies take into account migration cost. The selected host (node j' in cluster i') is the one with the minimun energy consumed with best execution time, weighed by the migration cost between the current position of the VM and the potential host. To reduce the load of an overloaded host, it begins to migrate the slowest task. Selection policy will choose the task that will stay the longest on the host.

This migration algorithm's goal is to minimize the energy. During the execution of the task it may happen that a node is overloaded. We decided in this case to migrate VMs whose execution time remaining is greater.

Policy of localization will then identify the host that will receive the task without exceeding its capacities (ie. its load after migration will still be under ε). So this host will be the

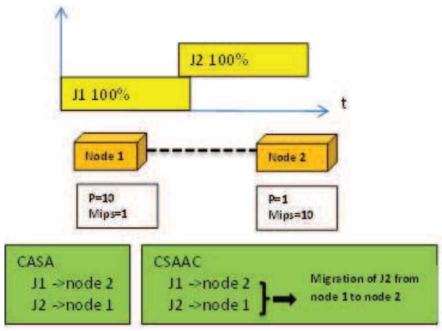

Fig. 2. How migrations can reduce makespan. CSAAC compared to CASA

new destination of the task. Figure 2 shows that CSAAC can produce better execution time due to migration.

V. Experiments and Results

In order to evaluate the gains of CSAAC compared to classical algorithms, we implemented this algorithm in our simulator based on MaGateSim [7], with the addition of power consumption and virtual machine (mainly their migration). It is designed to be a decentralized grid scheduler that emphasizes on cloud scheduler interoperation, and complemented by a dynamic resource discovery approach on decentralized network. In order to share the jobs submitted from a local CSAAC to other CSAAC within the same grid community, a set of community scheduling relevant parameters are evaluated and discussed to address various job delegation scenarios between different CSAAC. CSAAC schedulers are driven to cooperate with each other, to provide intelligent scheduling for the scope of serving the grid community as a whole, not just for a single grid node individually. Our simulator is a java event driven simulator of Clouds. It provides information about execution times, but also about instantaneous power consumption and energy consumption of tasks. We have also performed an implementation of CASA within the context of current developments in cloud computing.

A. *Simulation environment*

The workload trace archive and resource deployment topology of the Grid5000 [9] is selected to organize the experiment of this work. We use 1700 jobs of grid5000 workload.The grid consists of 9 sites, 26 nodes and 3194 processors. Hosts have two different power states for each core : Switched on and switched off. While switched on, the power consumption depends on load, P_{min} and P_{max}. Those values are different for each host and are respectively between 75 and 150W, and 200 and 560W as measured on Grid5000. In the following we compare CSAAC with algorithms CASA.

B. *Experimental results*

In this subsection, we describe the simulation study performed to evaluate the performance of our algorithms in terms of energy minimization as well as the execution time and the number of migrations. We compare our algorithms with the CASA [10] algorithm that produces energy-efficient schedules. The first observation is that for two algorithms, CSAAC

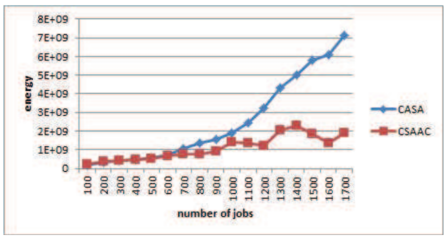

Fig. 3. Energy of algorithms compared to CASA with sorted hosts by pmax. Lower is better

consumes the least energy while CASA algorithm consumes the most energy (see figure 3), when the number of jobs $T > 300$. For a small number of tasks our algorithm leads to a significant energy consumption. The second observation is that CSAAC algorithm is able to reduce the energy consumption by 5 percent to 80 percent when job increases from 300 to 1700. Figure 4 demonstrates the energy consumption and the number of migrations incurred by 1700 jobs. An obvious observation is that migrations are beneficial to save energy. The second phase of our algorithm calls into question the choices and therefore modifies the host loads over time. If at any time the tasks are finished and that there's several machines under loaded, you can consolidate them. This is not the case for CASA which never calls into question the allocation once the jobs is running. In addition we can't use jobs between neighbor and find a more efficient than neighboring nodes starting from step by step. The impact of migration, however, may not be large enough to dominate the total energy consumption when the number of jobs is less than 300. As CSAAC has more flexibility with

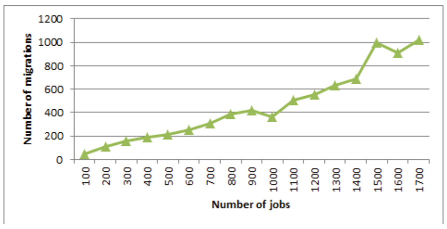

Fig. 4. CSAAC : Migration

using the migration possibilities, simulation showed that for all cases, maskespan of jobs with CSAAC is always lower than with CASA.

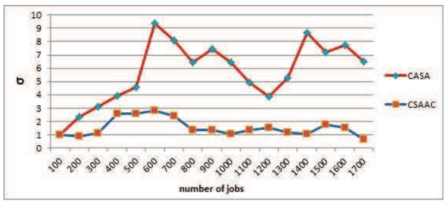

Fig. 5. comparison between two algorithms CSAAC and CASA. Standard deviation σ of node load

Due to the thresholds of CSAAC, it would be possible to reduce further the number of switched on hosts but it would overload remaining hosts. Those hosts would become hot points and would have a negative impact on cooling. In order to prevent overloading, CSAAC adjusts load. Also the figure 5 show how widely host load are dispersed from the average value (the mean). In previous results (figures 3 and 4),the CSAAC algorithm is in the lead in terms of energy gain and execution time, since in this algorithm there is cooperation between schedulers which allows an efficient consolidation in cloud. Figure 5, shows the standard deviation σ of the nodes load which confirms the good distribution of the load after consolidation. Thus figure 5 shows that the algorithm CSAAC gives the best standard deviation after compared CASA, which is an indication of the good predictability of the performance of the algorithm CSAAC. We observe that when the number of tasks increases, our algorithms perform better.

VI. Conclusion

In this paper, we presented and evaluated our energy-efficient migration algorithm for cloud. This algorithm is based on the principle of *CASA*. It takes into account energy and provides energy-efficiency improvement compared to classical load unbalancing algorithms. Also current version of CSAAC is decentralized.

Our main problem was to optimize energy consumption given task performance constraints. Energy consumption is to be taken in a broad way as we try to prevent hot spots to reduce impact on cooling.

Thus, we have compared CSAAC to CASA over a range of realistic problem instances using simulation. CSAAC parameters lead to a family of heuristics that perform well in terms of energy savings while still leading to good task performance. It consolidate tasks on a subset of the cluster hosts judiciously chosen depending on the characteristics and state of resources. This algorithm has a low computational cost. It can then be employed in practical settings. Overall, the proposed CSAAC algorithm can compute allocations effectively with an important energy gain. The simulations in this paper were based on heterogeneous processors with a real job workload, but it des not take into account latency and energy consumed during migration and communication. However, our proposed algorithms can be easily extended to those systems. Our simulator which extends MaGateSim is selected as the simulator for experimental evaluation. The first phase of this work was to extend MaGateSim by adding properties that allow it to take into account energy and migration. We have also performed an implementation of CASA within the context of current developments in cloud computing.

The simulation results showed that our algorithm is capable of obtaining energy-efficient schedules using less optimization time. In particular, we showed that for the case when $jobs > 300$, our algorithm is able to reduce the average energy consumption by about 10 percent to 80 percent. At the same time, the execution time of jobs is also reduced by 5 percent to 25 percent when number of $jobs > 300$, when compared to the CASA algorithm. Finally, the encouraging results observed from this work serve as the motivation to improve the quality of service.

References

[1] Etinski, M., Corbalan, J., Labarta, J., Valero, M.: Utilization driven power-aware parallel job scheduling. Computer Science - Research and Development 25, 207-216 (2010), doi:10.1007/s00450-010-0129-x

[2] Lawson B., Smirni, E.: Power-aware resource allocation in high-end systems via online simulation. In: Proceedings of the 19th Annual international Conference on Supercomputing, ICS 2005, pp. 229-238. ACM, New York (2005)

[3] Rajkumar Buyya and Manzur Murshed, GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[4] R. Ranjan, A. Harwood, R. Buyya, SLA-based coordinated superscheduling scheme for computational Grids, in: 2006 IEEE International Conference on Cluster Computing, 2006, pp. 18.

[5] R. Ranjan, A. Harwood, R. Buyya, SLA-based coordinated superscheduling scheme for computational Grids, in: Cluster Computing, 2006 IEEE International Conference on, IEEE, 2007, pp. 18.

[6] R. Smith, The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Transactions on Computers 100 (29) (1980)11041113.

[7]] Y. Huang, A. Brocco, M. Courant, B. Hirsbrunner, P. Kuonen, MaGate Simulator: a simulation environment for a decentralized grid scheduler, Advanced Parallel Processing Technologies (2009) 273287.

[8] C. Comito et al, Energy Efficient Task Allocation over, Mobile Networks Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on, 12-14 Dec. 2011

[9] GWA, Grid5000 workload trace archive, http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Workloads.Gwa-t-2, 2010.

[10] Y. Huang et al, Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm, Future Generation Computer System (2011), doic : 10,.1016/j.future.2011.05.006

[11] 2009] Yu J. and Buyya R. Gridbus Workflow Enactment Engine, Grid Computing: Infrastructure, Service, and Applications, L. Wang et al. (eds.). CRC Press, USA, 2009.

[12] Ranjan R., Rahman M., and Buyya R. A decentralized and cooperative workflow scheduling algorithm. In Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid08), France, May, 2008.

[13] J. Kaplan, W. Forrest, N. Kindler, Revolutionizing Data Center Energy Efficiency, McKinsey Company, Tech. Rep.

[14] R. Buyya, A. Beloglazov, J. Abawajy, Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges, in: Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2010, Las Vegas, USA, 2010.

[15] E. Pinheiro, R. Bianchini, E.V. Carrera, T. Heath, Load balancing and unbalancing for power and performancee in cluster-based systems, in: Proceedings of the Workshop on Compilers and Operating Systems for Low Power, 2001, pp. 182195.

[16] Anton Beloglazov and Rajkumar Buyya, Energy Efficient Allocation of Virtual Machines in Cloud Data Centers, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing

[17] L. Chiaraviglio, I. Matta, GreenCoop: cooperative green routing with energyefficient servers, in: Proceedings of the 1st ACM International Conference on Energy-Efficient Computing and Networking, e-Energy 2010, Passau, Germany, 2010, pp. 191194.

[18] M. Koseoglu, E. Karasan, Joint resource and network scheduling with adaptive offset determination for optical burst switched grids, Future Generation Computer Systems 26 (4) (2010) 576589.

[19] Dalibor Klusek and Hana Rudov. Alea 2 - Job Scheduling Simulator. In proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010), ICST, 2010

[20] M. Murshed, R. Buyya, D. Abramson, GridSim: A Toolkit for the Modeling and Simulation of Global Grids, Technical Report, Monash-CSSE 2001/102, Monash University, Australia, November 2001.

[21] https://groups.google.com/forum/#!$forum/cloudsim$