International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)

International Conference on Information Systems and Computing (ICISC-2013), INDIA.

PROFIT MAXIMIZATION FOR SAAS USING SLA BASED SPOT PRICING IN CLOUD COMPUTING

N. Ani Brown Mary

(M.E) Computer Science, Anna University of Technology, Tirunelveli, India.

anibrownvimal@gmail.com

Abstract

Cloud Computing provides user a complete software environment. To serve resources to customers, Software as a Service providers rent resources from the Infrastructure as a Service Providers. Client applications and service providers negotiate for the sales of services by means of Service Level Agreement (SLA) that acts as a contract between them. Depending on the status of the demand, Infrastructure as a Service provider is able to offer higher or lower prices for maximising its profit. So it is difficult to establish a profitable pricing function for Software as a Service Providers. Therefore, this paper proposes spot pricing strategy to estimate the bid according to the used specified constraints of the user and also optimizing the pricing level to improve the profit of Saas providers. We also have analyzed the performance of Spot Pricing Scheme with different Scenario to maximize the Saas providers profit.

***Keywords*-- Cloud Computing; Software as a Service; Service Level Agreement; Infrastructure as a Service; Spot Pricing**

I. INTRODUCTION

Cloud computing can be viewed as the transformation into reality of a long held dream called ‘‘Computing as Utility’’, it emerged into the market with a huge potential to fulfill this dream. It promises on-demand services for a customer’s software, platform and infrastructure needs. In its fold, companies do not even need to plan for their IT growth in advance with this new ‘‘pay as you go’’ system. Already, there has been upbeat assessment about its great potential for utility, scalability and instant access features; but on the flip side, some are also apprehensive of security gaps involving for instance, trust, threats and risks. Cloud computing has emerged as a new paradigm for delivery of applications, platforms,or computing resources(processing power/ bandwidth/ storage) to customers. The Cloud model is cost-effective because customers pay for their actual usage without upfront costs, and scalable because it can be used more or less depending on the customers’ needs.A set of applications managed and hosted externally by a specialist partner and it is delivered over a secure high quality network it is also available anywhere with an internet connection, even when on the move. Cloud computing is an internet technology that utilizes both central remote servers and internet to manage the data and applications. This technology allows many businesses and users to use the data and application without an installation. Users and businesses can access the information and files at any computer system having an internet connection. Cloud computing provides much more effective computing by centralized memory, processing, storage and bandwidth.

‘‘Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services’’.A cluster of computer hardware and software that offer the services to the general public (probably for a price) makes up a ‘public cloud’.

In this paper, we focus on the SaaS layer,which allows customers to access applications over the Internet without software related cost and effort (such as software licensing and upgrade).The general objective of SaaS providers is to minimize cost and maximize customer satisfaction level (CSL). The cost includes the infrastructure cost, administration operation cost and penalty cost caused by SLA violations.CSL depends on to what degree SLA is satisfied.In general, SaaS providers utilize internal resources of its data centres or rent resources from a specific IaaS provider. For example, Saleforce.com [11] hosts resources but Animoto [12] rents resources from Amazon EC2 [13]. In-house hosting can generate administration and maintenance cost while renting resources from a single IaaS provider can impact the service quality offered to SaaS customers due to the variable performance [14].There are several factors that can affect an IT solution or managed service provider’s pricing methodology for Cloud Computing Services, including their predominant business model, the services that comprise their Cloud offerings, their costs to deliver these services, their sales sophistication, and their efficiencies in delivering these services. The less efficient they are at delivering service, the higher their price points must be to maintain desired net profitability.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

II. System Model

A customer sends requests for utilizing enterprise software services offered by a SaaS provider, who uses three layers, namely application layer, platform layer and infrastructure layer, to satisfy the customer's request. The application layer manages all application services that are offered to customers by the SaaS provider. The platform layer includes mapping and scheduling policies for translating the customer's Quality of Service (QoS) requirements to infrastructure level parameters and allocating Virtual Machines (VMs) to serve their requests. The infrastructure layer controls the actual initiation and removal of VMs. The VMs can be leased from IaaS providers such as Amazon EC2 or private virtualized clusters owned by the SaaS provider. In both cases, the minimization of the number of VMs will deliver savings. The savings are greater when SaaS providers use the third party IaaS providers since no capital expenditure is required.

Currently, SaaS providers such as Compiere ERP provide an individual VM for each customer to maintain service level requirements in terms of response time and capacity. However, this causes wastage of hardware resources which results in high infrastructure cost since customers may not use complete VM capacity which is reserved to serve their requests.

A multi-tenancy approach can reduce the needed infrastructure, but care must be taken in providing access to resources so that Service Level Agreements (SLAs) are not violated.The current works in Cloud computing are focused mostly on maximizing the profit of IaaS providers, but works related to the SaaS provider considering SLAs are still in their infancy. Many works do not consider the customer driven management, where resources have to be dynamically rearranged based on customers' demands. Thus, in this paper, we examine the allocation strategies, which allow a cost effective usage of resources in Clouds to satisfy dynamically changing customer demands in line with SLAs. Therefore, in order to achieve the SaaS provider's objective to maximize profit and customer satisfaction levels, our work proposes spot pricing technique that is used to reduce VM's and increase Saas providers profit.

III. Maximizing Profit

In the existing systems they use various profit maximization algorithms such as Maximizing the profit by minimizing the number of VMs (ProfminVM) , Maximizing the profit by rescheduling (ProfRS) , Maximizing the profit by exploiting the penalty delay (ProfPD).In these algorithms they don't use the pricing strategies.After the process of negotiation directly scheduling algorithms will be executed.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

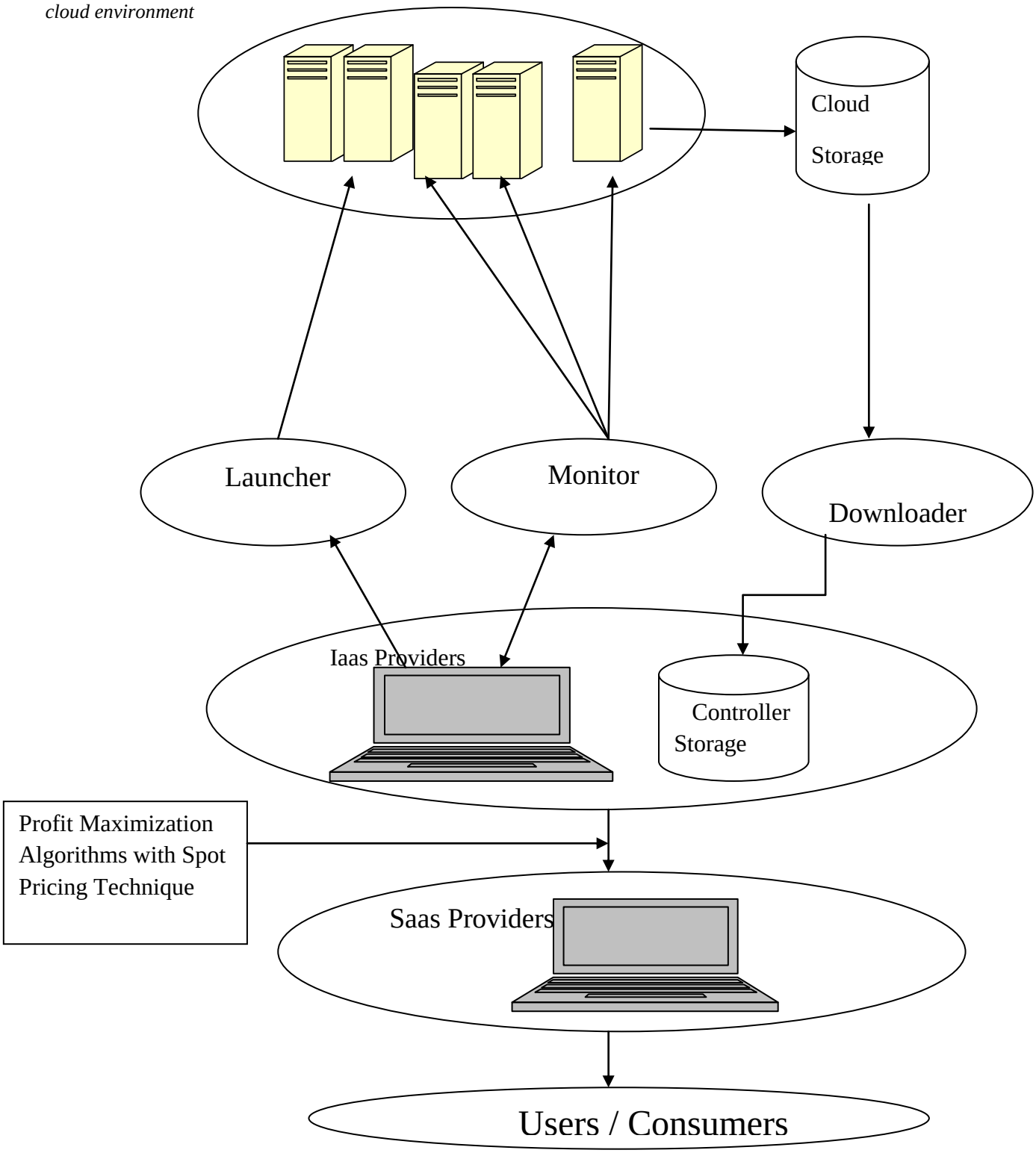

Fig 1: A System model of a Saas layer structure

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

The first process is to accept new user request, so we check if the new user request can be processed by queuing it up at the end of an already initiated VM or by inserting it into an initiated VM, or by initiating a new VM. Hence, this algorithm checks if the new request can wait all accepted requests to complete in any initiated VM. We first verifies each VM in each resource provider if the flexible time that is (fT^{new}_{ijl}) of the new request is enough to wait for all accepted requests in vm_{il} to complete.

$$fT^{new}_{ijl} = DL^{new} - \sum_{k=1}^{k} procT^{k}_{ijl} - SubT^{new} ; \forall i \in I, j \in J, k \in K, l \in N_j \quad ---1$$

$$T^{new}_{ijl} = \sum_{k=1}^{k} procT + procT^{new}_{ijl} , \text{ if new VM is not initiated} \quad ------2$$

$$T^{new}_{ijl} = procT^{new}_{ijl} + iniT_{ijl} + DTT^{new}_{ijl} , \text{ if new VM is initiated} \quad -----3$$

$$ret^{new}_{ijl} = (prof^{new}_{ijl}) / T^{new}_{ijl} ; \forall i \in I, j \in J, l \in N j \quad ---------4$$

Here K indicates total number of all accepted requests, I indicates all VMs, J indicates all resource providers, l indicates VM type, and N j indicates all VM types provided by resource provider j. If new request can wait for all accepted requests to complete, and then the investment return is calculated. If there is value added according to the investment return, and then all related information such as resource provider ID, VM ID, start time and estimated finish time are stored into the potential schedule list.

If this request cannot wait in any initiated VM, then the algorithm checks if it can be accepted by initiating a new VM provided by any IaaS provider. Here it first checks for each type of VMs in each resource provider in order to determine whether the deadline of new request is long enough comparing to the estimated finish time. The estimated finish time depends on the estimated start time, request processing request If a SaaS provider does not make any profit by utilizing already initiated VMs nor by initiating a new VM to accept the request, then the algorithm rejects the request.

Thus, the response time (T^{new}_{ijl}) for the new request to be processed on VM_{il} of IaaS provider j is calculated, and it consists of VM initiation time ($iniT^{new}_{ijl}$), request's service processing time ($procT^{new}_{ijl}$), data transfer time (DTT^{new}_{ijl}),and penalty delay time (PDT^{new}_{ijl}). The investment return (ret^{new}_{ijl}) to accept new user request per hour on a particular VMil in IaaS provider j is calculated based on the profit ($prof^{new}_{ijl}$) and time (T^{new}_{ijl})

Time and VM initiation time. If the new request can be completed within the deadline investment return is calculated.

Due to increase in demand for utilizing public Cloud resources, we are facing with many trade-offs between price, performance and recently reliability. Amazon's Spot Instances (SIs) provide a low price yet less reliable and competitive bidding option for the public Cloud users.

TABLE I
USER PARAMETERS AND CONSTRAINTS

Notation	Description
N_{inst}	Number of instances that process the work in parallel
N_{max}	Upper bound on ninst
W	Total amount of work in the user's job
W_{inst}	Workload per instance (W=ninst)
T	Task length, time to process Winst on a specific instance
B	Budget per instance
C_B	user's desired confidence in meeting budget B
T_{dead}	deadline on the user's job
C_{dead}	desired confidence in meeting job's deadline
U_b	user's bid on a Spot Instance type
I_{type}	EC2 instance type

IV. SPOT PRICING TECHNIQUES

Further constraints which might be specified as part of user's input are:

budget B: upper bound on the total monetary cost per instance

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

cB: user's desired confidence in meeting this budget

deadline tdead: upper bound on the execution time (clock time needed to process Winst)

cdead: the desired confidence in meeting tdead.

Note that there is a potential exploitation method to reducethe cost of the last partial hour of work called "Delayed Termination" [6]. In this scenario, a user waits after finished computation almost to the next hour-boundary for a possible termination due to an out-of-bid situation. This potentially prevents a payment for the computation in the last partial hour.

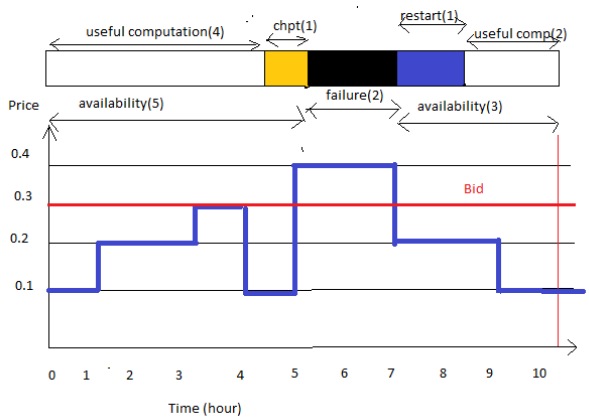

Fig. 2. Execution model of Spot pricing

V. Execution Scenario

The Grid and Desktop Grid workload parameters correspond to relatively small and relatively large jobs respectively. This is partly because Grids have on the order of hundreds to thousands or resources, and Desktop Grids have on the order of tens to hundreds of thousands of resources. So the workload size is reflective of the platform size. The specific Grid and Desktop Grid workload parameters that we use are based the BOINC Catalog [7] (Workload W1), and Grid Workload Archive [8], [9] (Workload W2), respectively (Table II).

Workload W1. In the BOINC Catalog [7], we find that the median job deadline tdead is 9 days, and the mean task length T is 4:6 hours (276 minutes) on a 2.5GHz core. This translates to a mean per instance workload Winst of 11:5 unit-hours. We will assume in the following an instance type with 2.5 EC2 Compute Units (e.g. a single core of the High-CPU medium instance type) [3] so that the task lengths remain around the original values. We also learned that a typical value for nmax is 20,000 tasks. Thus, we center the range of W, tdead, Winst (or equivalently, of T) around these values. See Table II for these and additional parameters.

Workload W2. From the Grid Workloads Archive [8], [9], we find that the mean job deadline tdead is 1074 minutes (17:9 hours), and the mean task length T is 164 minutes (2:7 hours) on a 2.5GHz core. This gives us an average perinstance workload Winst of 6:75 unit-hours. nmax is 50 tasks, the highest average reported in [9]. We will again assume in the following an instance type with 2.5 EC2 Compute Units for a single core.

Figure 2 illustrates an execution scenario. A user submits a job with a total amount of work W of 12 unithours with ninst = 2 which translates to a Winst = 6 unithours and the task time (per instance) T of 6 hours (assuming EC2's "small instance" server). User's bid price ub is 0.30 USD, and during the course of the job's computation, the job encounters a failure (i.e. an out-of-bid situation) between time 5 and 7. The total availability time was 8 hours, from which the job has (4+2) = 6 hours of useful computation, and uses 1 hour for checkpointing and 1 hour for restart. (Obviously, these overheads are unrealistic, but defined here for simplicity of the example.) The clock time needed until finishing was 10 hours. During the job's active execution, the spot price fluctuates; there are 3 hours at 0:10 per time unit, 4 hours at 0:20 per time unit, and 1 time unit at 0:30 per time unit, giving a total cost of 1:40. Thus the expected price is 1:40=8 = 0:175 (USD / hour).

VI. Modeling Of The Execution

The execution is modeled by the following random variables (RVs):

Eexecution time ET is the total clock time needed to process Winst on a given instance (or, equivalently, to give the user T hours of useful computation on this instance); in the example, ET assumes the value of 10 hours.

Availability time AT is the total time in-bid; in our example, this is 8 hours.

Expected price EP is the total amount paid for this instance to perform Winst divided by the total availability time; note that always EP _ ub.

Monetary cost M is the amount to be payed by the user per instance, defined by M = AT _ EP; in the example, we have (in USD) M = 8 _ 0:175 = 1:40.

Note that as we assume ninst instances of the same type, they all are simultaneously in-bid and out-of-bid; therefore, the values of the variables ET, AT, EP are identical for all instances deployed in parallel. In particular, the whole job completes after time ET, and so ET is also the job's execution time. Furthermore, all the above RVs depend on the checkpointing strategy.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

As designing and optimizing such a strategy is a complex undertaking beyond the scope of this work, we assume here one of the following two simple strategies taken from [6]:

OPT - the optimal strategy, where a checkpoint is taken just prior to a failure; this (unrealistic) strategy serves as a base-case for cost comparison.

HOUR - the hourly strategy, where a checkpoint is taken is at a boundary of each paid hour (measured from the start of current availability interval).

VII. Models Independent Of The Task Time T

The following RVs give an alternative characterization of the execution process:

Availability ratio AR = AT=ET is the ratio of the total availability time (time in-bid) to execution time; in our example, AR = 8=10.

Utilization ratio UR is T=ET, or ratio of the total useful computation to the execution time; in the example, UR = 6=10. This approach requires to store only the distributions of AR and UR for representative pairs (bid price ub, instance type); the independence of the the task length T saves a lot of storage and simulation time. However, our experimental findings show that both AR and UR depend significantly on T. Thus, to optimize accurately, we need to store the distributions of these RVs for many values of T. In face of this effect we decided to simulate and store directly the distributions of RVs.

VIII. Related Work

Research on market driven resource allocation was started in early 80s [2][3]. Most of the market-based resource allocation methods are either non-pricing-based [10] or designed for fixed number of resources, such as FirstPrice [4] and FirstProfit [5]. Our work is related to user driven SLA-based profit maximization resource allocation for SaaS providers with Spot Pricing.

Yi et. al. in [15] introduced some checkpointing mechanisms for reducing costs of Spot Instances. They used the real price history of EC2 Spot instances, and show how the adaptive checkpointing schemes are able to decrease the monetary cost and improve the job completion times.

In [16], a decision model for the optimization of performance, cost and reliability under SLA constrains is proposed. They used the real price history and workload models, to demonstrate how their proposed model can be used to bid optimally on Spot Instances to reach different objective with desired levels of confidences.

Chohan et. al. in [17] proposed a method to utilize the Spot Instances to speed up the MapReduce tasks. They provide a Markov Chain to predict the probability of the Spot Instance lifetime. They concluded that having a fault tolerant mechanism is essential to run MapReduce jobs on Spot Instances.

In [18], they proposed a hybrid cloud architecture to lease the Spot Instances to manage peak loads of a local cluster. They proposed some provisioning policies and investigate the utilization of Spot Instances compared to on-demand instances in terms of monetary cost saving and number of deadline violations.

IX. Experimental Results

We use CloudSim as a Cloud environment simulator and implement our algorithms within this environment. We observe the performance of the proposed algorithms from both users' and SaaS providers' perspectives. From users' perspective, we observe how many requests are accepted and how fast user requests are processed (we call it average response time). From SaaS providers' perspective, we observe how much profit they gain and how many VMs they initiate. It shows how the spot pricing is successful for all users and it is satisfied by both saas providers and users.

X. Conclusion

Thus the spot pricing technique is used for bidding process between users so that it increases the profit of software as a service providers .Compared to other pricing techniques spot pricing is the most effective technique.

Acknowledgement

None of this work would have been possible without the selfless assistance of a great number of people. I would like to gratefully thank all those members for their valued guidance, time, helpful discussion and contribution to this work.

REFERENCES

[1] Linlin Wu ∗, Saurabh Kumar Garg, Rajkumar Buyya," SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments", published in Journal of Computer and System Sciences 78 (2012) 1280–129.

[2] D. Parkhill, "The challenge of the computer utility", 1966, Addision- Wesley Educational Publishers Inc., USA.

[3] Y. Yemini, "Selfish optimization in computer networks processing". In Proceeding of the 20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes, San Diego, USA.

[4] O. F. Rana, M. Warnier, T. B. Quillinan, F. Brazier, and D. Cojocarasu, "Managing Violations in Service level agreements". In proceedings of the 5th International Workshop on Grid Economics and Business Models (GenCon 2008), Gran Canaris, Spain.

[5] I. Popovici, and J. Wiles, "Proitable services in an uncertain world". In Proceeding of the18th Conference on Supercomputing (SC 2005), Seattle, WA.

[6] S. Yi, D. Kondo, and A. Andrzejak, "Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud," March 2010.

[7] "Catalog of boinc projects," http://boinc-wiki.ath.cx/index.php?title= Catalog_of_BOINC_Powered_Projects.

[8] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema, "The grid workloads archive," Future Generation Comp. Syst., vol. 24, no. 7, pp. 672–686, 2008.

[9] A. Iosup, O. O. Sonmez, S. Anoep, and D. H. J. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in HPDC, 2008, pp. 97–108.

[10] J. Broberg, S. Venugopal, and R Buyya, Market-oriented Grids and Utility Computing: The state-of-the-art and future directions, Journal of Grid Computing, 3(6), (pp.255-276).

[11] Saleforce.com, retrieved on 10 Sep. 2010, http://www.salesforce.com/au/.

[12] Animoto, retrieved on 12 Sep. 2010, http://developer.amazonwebservices.com.

[13] J. Varia, Architecting applications for the Amazon Cloud, in: R. Buyya, J. Broberg, A. Goscinski (Eds.), Cloud Computing: Principles and Paradigms, Wiley Press, New York, USA, ISBN: 978-0470887998, 2010,http://aws.amazon.com.

[14] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, A performance analysis of EC2 cloud computing services for scientific computing, in: Proceedings of 1st International Conference on Cloud Computing (CloudComp), Munich, Germany, 2009.

[15] Sangho Yi, Derrick Kondo, and Artur Andrzejak. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In 3rd IEEE International Conference on Cloud Computing, pages 236 –243, 2010.

[16] Artur Andrzejak, Derrick Kondo, and Sangho Yi. Decision model for cloud computing under SLA constraints. In 18th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pages 257–266, 2010.

[17] Navraj Chohan, Claris Castillo, Mike Spreitzer, Malgorzata Steinder, Asser Tantawi, and Chandra Krintz. See spot run: using spot instances for MapReduce workflows. In the 2nd USENIX conference on Hot topics in cloud computing, HotCloud'10, pages 7–7, 2010.

[18] Michael Mattess, Christian Vecchiola, and Rajkumar Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market.In 12th IEEE International Conference on High Performance Computing and Communications ,pages180–188, 2010.