Global Journal of Computer Science and Technology
Cloud and Distributed
Volume 13 Issue 1 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Application Deployment Models with Load Balancing Mechanisms using Service Level Agreement Scheduling in Cloud Computing

By Prof. Jasobanta Laha, Prof. (Dr.) R. N. Satpat & Prof. (Dr.) C. R. Panda
Suddhananda Engineering & Research Centre, Bhubaneswar

Abstract - Cloud computing is a market-oriented computing paradigm with virtually unlimited scalable high performance computing resources. The High level middleware services for cloud computing and cloud workflow systems are research frontier for both cloud computing and workflow technologies. In this paper, the extension of Cloud management infrastructure with Service Level Agreement (SLA) aware application and motivating scenario for the development of the scheduling heuristic after which, the detail design and implementation of the heuristic are mentioned.

Keywords : *qos, SaaS, SLA, lom2his, cloudsim, vm, iaas, PaaS.*

GJCST-B Classification : *C.2.5*

Strictly as per the compliance and regulations of:

Application Deployment Models with Load Balancing Mechanisms using Service Level Agreement Scheduling in Cloud Computing

Prof. Jasobanta Laha [α], Prof. (Dr.) R. N. Satpat [σ] & Prof. (Dr.) C. R. Panda [ρ]

Abstract - **Cloud computing is a market-oriented computing paradigm with virtually unlimited scalable high performance computing resources. The High level middleware services for cloud computing and cloud workflow systems are research frontier for both cloud computing and workflow technologies. In this paper, the extension of Cloud management infrastructure with Service Level Agreement (SLA) aware application and motivating scenario for the development of the scheduling heuristic after which, the detail design and implementation of the heuristic are mentioned.**

Keywords : qos, SaaS, SLA, lom2his, cloudsim , vm, iaas, PaaS.

I. Introduction

Cloud Computing is a computing paradigm [2, 6] that refers to variety of services available on internet which delivers computing functionality on the service provider's infrastructure. Cloud is a pool of virtualized computer resources and may be hosted on grid or utility computing environments [1, 7]. Its potential feature which includes the ability to scale, to meet changing users demand, separation of infrastructure maintenance duties from users, location of infrastructure areas with electricity, sharing of peak load capacity and so on. In this modern technical ear, the recent popularity of cloud computing on the scenario of data and computation and investigation of intensive scientific workflow applications like, climate modeling, earthquake modeling, weather forecasting, disaster recovery simulation, Astrophysics and high energy physics [5, 8, 9]. These scientific processes can be modeled or redesigned as scientific cloud workflow specifications at build-time modeling stage [5]. These specifications contain number of data computation activities and their non-functional requirements such as Quos constraints on time and cost basis [10]. At runtime execution stage, with the support of cloud workflow execution functionalities such as workflow scheduling [11], load balancing [3] and temporal verification [4], cloud workflow instances are executed by employing the time and cost basis [10]. At runtime execution stage, with the support of cloud workflow execution functionalities such as workflow scheduling [11], load balancing [3] and temporal verification [4], cloud workflow instances are executed by employing the supercomputing and data sharing ability computing infrastructures with satisfactory Quos.

Scientific applications are time constrained which required to be completed by satisfying a set of temporal constraints and global temporal constraints. The task execution time is the basic measurements for system performance, which need to be monitored and controlled by specific system management mechanisms. To ensure satisfactory temporal correctness and on time completion of workflow application is a critical issue for enhancing the overall performance. Cloud customers are interested in cost effective deployment of single applications in clouds, which is common in the Software as a Service (SaaS) delivery model. Commercial cloud providers such as sales force [24] are offering the provision for single applications based on agreed SLA terms. Commercial providers use custom techniques, which are not open to the general public. To foster competitive cloud market and reduce cost, the need for open solutions is must.

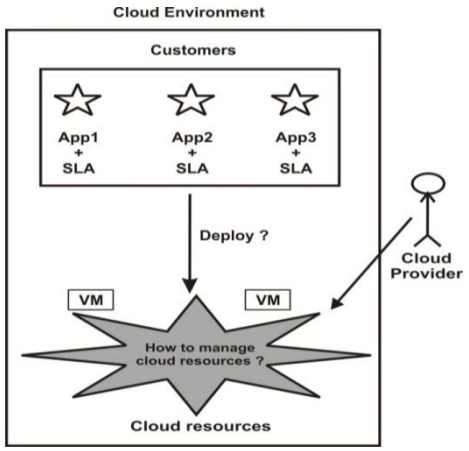

Figure 1 : Scheduling Motivation Scenario

In Figure – 1, a motivating scenario for the development of scheduling and deployment heuristic is represented. The use case scenario shows a Cloud provider and pools of customers who wish to deploy

Author α : Pro Associate Professor SERC, Bhubaneswar.
E-mail : yeshmcp@gmail.com
Author σ : Principal, HIT, Bhubaneswar.
E-mail : rabi_satpathy@yahoo.co.in
Author ρ : Principal, SERC, Bhubaneswar.
E-mail : panda_chitaranjan@rediffmail.com

their applications on the Cloud resources based on the pre agreed SLA objectives. The application of how to deploy the available virtual machines in the Cloud to ensure their performance and enforce the agreed SLA objectives including the challenge to manage resources to achieve high utilizations and maximum deployments by grid scientific computing [17]. A novel scheduling heuristic considering multiple SLA parameter objectives, such as amount of required CPU, network bandwidth, and storage in deploying applications on Clouds is mentioned. The heuristic includes a load-balancing mechanism for efficient distribution of the applications' execution on the Cloud resources. We also present a flexible on-demand resource allocation strategy included in the heuristic for automatically starting new virtual machines (VM) when a non-appropriate VM is available for application deployment. The concept and detailed design of the heuristic including its implementation in the Clouds simulation tool [15, 16]. The scheduling strategy proposed in this section is integrated into the LoM2HiS framework as shown in Figure - 2.

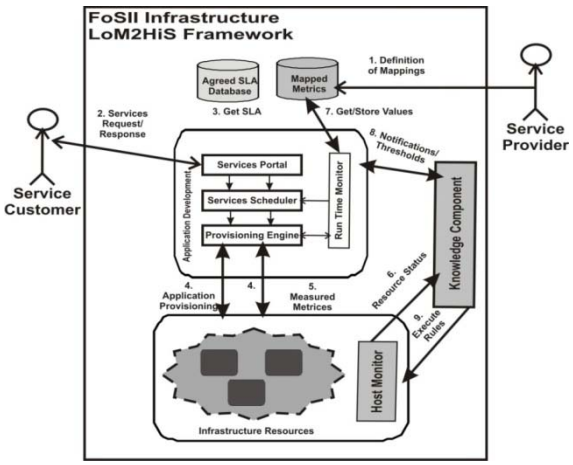

Figure 2 : LoM2HiS with Scheduling Strategy

II. Resource Provisioning

The idea of Cloud computing is to provide resources as a service in a flexible and scalable manner [14]. There are three well known types of resource provisioning [21, 23] in Cloud:

i. Infrastructure as a Service (IaaS) that offers bare hardwires such as the physical machines, storage devices, and network infrastructures as a service. Amazon EC2 [12] is an example of IaaS offering;
ii. Platform as a Service (PaaS), which delivers platform for application development and deployment as a service [20]. It typically utilizes virtualized resources in form of virtual machines that are capable of provisioning resources to applications. A typical example of PaaS service is the Google App Engine [20];
iii. Software as a Service (SaaS) offering resources for the provisioning of single Applications in a Cloud environment. Vendors of SaaS include salesforce.com [24].

The Cloud provisioning and deployment model presented in Figure - 3 shows a scenario combining the three different types of resource provisioning to host service requested from customers. The customers place their service deployment requests to the service portal, which passes the requests to the request processing component to validate the requests. If the request is validated, it is then forwarded to the scheduler. The scheduler selects the appropriate VMs through the provisioning engine in PaaS layer for deploying the requested service and the load balancer balances the service provisioning among the running VMs. The provision engine manages the VMs on the virtualization layer and the virtualization layer interacts with the physical resources via the provision engine in IaaS layer. The low-level resource metrics of the physical resources at the IaaS layer are monitored by the LoM2HiS framework [19]. The service status and the SLA information are communicated back to the service portal [22]. Although, the scenario described above is a possible way of combining the three types of resource provisioning, there exist other scenarios like provisioning of virtual machines alone and provisioning of physical resources alone, which are possibilities of provisioning at the single layers alone.

However, our approach aims to provide an integrated resource provisioning strategy. The proposed scheduling heuristics considers the three layers. Efficient resource provisioning and application deployments at these layers are not trivial considering their different constraints and requirements. At the IaaS layer the physical resources must be managed to optimize utilizations. At the PaaS layer, the VMs have to be deployed and maintained on the physical host considering the agreed SLAs with the customer.

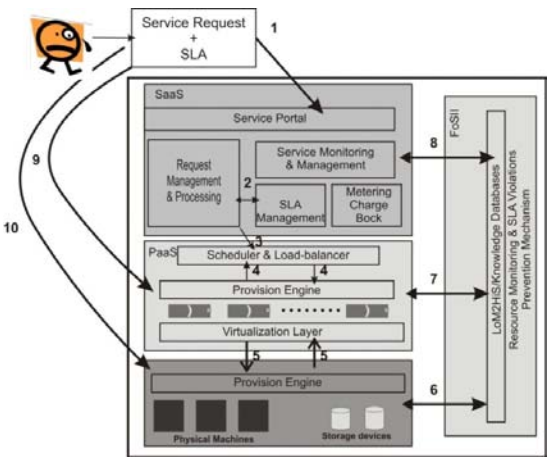

Figure 3 : Cloud Provisioning and Deployment Model

III. Scheduling and Load Balancing Mechanisms

The proposed scheduling heuristic [18] aims at deploying applications on virtual machines based on the agreed SLA objectives. Moreover, the integrated load-balancer in the heuristic ensures high and efficient resource utilization; consequently saving the provider the cost of maintaining unused resources. In this work, we assume that the SLA terms between the customer and the Cloud provider are already established. Thus, the processes of SLA specification, negotiation, and establishment are out of scope for this work, but there is ongoing research work where the Vie SLAF framework [13] is used to address the issues.

Algorithm - 1:
(Scheduling Heuristic)

1. Input: User Service Request
2. Get Global Resources and Available Vm List;
3. // find appropriate VM List
4. If AP(R, AR)! =; then
5. //call the load balancing algorithm
6. deployable VM = load-balance (AP(R, AR))
7. deploy service on deployable Vm;
8. deployed = true;
9. else
10. If global Resource Able to Host Extra VM then
11. Start new VM Instance;
12. Add VM to Available VM List;
13. Deploy service on new Vm;
14. Deployed = true;
15. Else
16. Queue Service Request Until
17. Queue Time > Waiting Time
18. Deployed = False;
19. End if
20. End if
21. If Deployed then
22. Return Successful;
23. Terminate;
24. Else
25. Return Failure;
26. Terminate;
27. End if

By the pseudo code presented in Algorithm - 1, the scheduler receives as input the customers' service requests (R) that are composed of the SLA objectives (S) and the application data (A) to be provisioned (line - 1 in Algorithm - 1). The request can be expressed as R = (S, A). Each SLA agreement has a unique identifier id and a collection of SLA Objectives (SLOs). The SLOs can be defined as predicates of the form:

$$SLO_{id}\left(x_i, comp, \pi_i\right) \text{with comp} \in \{<, \leq, >, \geq, =\} \quad (1)$$

Where x_i € {Bandwidth, Memory, Storage, Availability} represents sample SLA parameters, comp the appropriate comparison operator, and π_i the values of the objectives.

The basis for finding the virtual machine with the appropriate resources for deploying the application, gathers the output of the scheduler and the confirmation about successful deployment or error message in case of failure. In first step, it extracts the SLA objectives information about the total available resources (AR) and the number of running virtual machines in the data center (line - 2). The SLA objectives are used to find a list of appropriate virtual machines (AP) capable of provisioning the requested service (R). This operation can be expressed as:

$$AP(R, AR) = \{VM : \text{VM} \in \text{AR}, \text{capable}(\text{VM}, \text{R})\} \quad (2)$$

Where capable (VM, R) is a predicate that returns true if the virtual machine is capable of provisioning the particular request or false otherwise (lines 3-4). Once the list of VMs is found, the load-balancer decides on which particular VM to deploy the application in order to balance the load in the data center (lines 5-8).

In case no VM, the appropriate resources running in the data center, the scheduler checks if the global resources consisting of physical machines can host new VMs (lines 9-10). If that is the case, it automatically starts new VMs with predefined resource capacities to provision service requests (lines 11-14). When the global resources cannot host extra VMs, the scheduler queues the provisioning of service requests until a VM with appropriate resources is available (lines 15-16). If after a certain period of time, the service requests cannot be scheduled and deployed, the scheduler returns a scheduling failure to the Cloud admin, otherwise it returns success (lines 17-27). The load-balancer shown in Algorithm - 2 is not an extension of Next-Fit algorithm and has two core differences like:

i. It does not fill a box to the full before starting to fill another one and
ii. It goes back to the half filled boxes to add new items.

The similarity lays in each iteration, does not put items in the last filled box unless there is no other appropriate box among all the boxes. In Algorithm - 2, the load balancer receives as input the appropriate VM list (line - 1 in Algorithm - 2). It first gets the number of available running VMs in the data center in order to

know how to balance the load among them (line - 2). Then it gets a list of used VMs, i.e., VMs that are already provisioning applications (line - 3). If this list is equal to the number of running VMs, it clears the list because all the VMs are currently provisioning some applications (lines 4-7). The first VM from the appropriate VM list can be selected for the deployment of the new application request. The selected VM is then added to the list of used VMs so that the load -balancer does not select it in the next iteration (lines 8-15).

The load-balancer tries to place one application on each VM running in the data center in the first phase after which it goes back again to place new applications on the VMs. The idea is that VMs executing less number of applications perform better than ones executing many applications while the others are running empty. The load-balancer alone has a total worst-case complexity of $0(n^2)$ in load balancing and selecting the specific VM for application deployment. This worst-case complexity is attributed by two processes:

i. By the processes of selecting the specific VM, which has a worst-case complexity of 0(n) because the load balancer in worst case has to go through the appropriate VM list of n size to select a specific VM
ii. By the processes of balancing the load among the VMs, which has a worse-case complexity of 0(n).

The Algorithm - 2 shows lines 8-14, this process is a sub-process of selecting the specific VM. Thus, the total worst-case complexity is of $0(n^2)$.

Algorithm – 2:

(Load Balancing Strategy)

1. Input: AP(R, AR)
2. Global Variable Available Vm List
3. Global Variable Used Vm List;
4. Deployable Vm = null;
5. If Size (Used VM List) == Size (Available Vm List) then
6. Clear Used Vm List;
7. End if
8. For vm in AP(R, AR) do
9. If vm not in Used Vm List thens
10. Add vm to Used Vm List;
11. Deployable Vm = vm;
12. Break;
13. End if
14. End for
15. Return Deployable Vm;

The scheduling heuristic without the load-balancer has a worst-case complexity of 0(m + n). This complexity is defined by the processes of finding out the resource capacities of the m physical machines and n available virtual machines in the data center. Other operations of the heuristic have constant complexity (0(1)) except the process of checking the available resources on the physical machines in order to start new VMs, which has a worst-case complexity of 0(m).

The total worst-case complexity of the proposed heuristic is a result of the sum of the scheduling heuristic complexity and the load-balancer complexity expressed at run time is:

$$0(m+n)+0(n^2)=0(n^2+m) \quad (3)$$

IV. Implementation

The proposed scheduling heuristic is implemented as a new scheduling policy in the Clouds simulation tool for the purpose of evaluation offering with unique features are:

i. Support for modeling and simulation of large scale Cloud computing environments including data centers, on a single computing machine
ii. A self-contained platform for modeling Clouds, service brokers, resource provisioning and application allocation policies
iii. Capability of simulating network connections among simulated components
iv. Support for simulation of federated Cloud environment able to network resources from both private and public providers.
v. Availability of virtualization engine that aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center's physical machine
vi. Ability to switch between spaces shared and time-shared allocation of CPU cores to virtualized resources.

Clouds components shown in the custom extensions layer in Figure – 4, has infrastructure level services as modeled by the core layer representing the original Clouds data center, with homogeneous or heterogeneous configuration of their hardware. Each data center instantiates a resource provisioning component that implements a set of policies that allocates resources to computing host and virtual machines. The two groups of Java classes in clouds are:

i) The control policy classes
ii) The simulation specific classes

The control policy classes include the implementations of a new data center broker for interfacing with the data center and our proposed scheduling heuristic. The data center broker is responsible for mediating negotiations between customers and Cloud providers in respect of allocating appropriate resources to customer services to meet their application's Quos needs and to manage the provider resources in the Clouds.

The extended data center broker includes the capability of running dynamic simulations by removing the burden of statically configuring the whole simulation scenario before starting .With this feature one can

generate and send in new events (service requests) during the simulation runtime.

The proposed scheduling heuristic provides policies used by the data center broker for allocating resources to applications. The implementations of the heuristic and that of the load balancer are realized with Java methods in a class named SLA Aware Scheduler as shown in Figure - 4. This class is used by the Data center Broker class to schedule, deploy applications, and manage the data center resources.

The simulation specific classes are used in realizing simulation scenarios. This group includes two Java classes named Data center Model and Service Request as shown in Figure - 4. Data center Model class presents methods for flexible instantiation of different data center scenarios for scalable simulations. The Service Request class represents a customer service request. It encapsulates information about the SLA parameter objectives agreed between the customer and the provider and the application data to be provisioned in the Cloud.

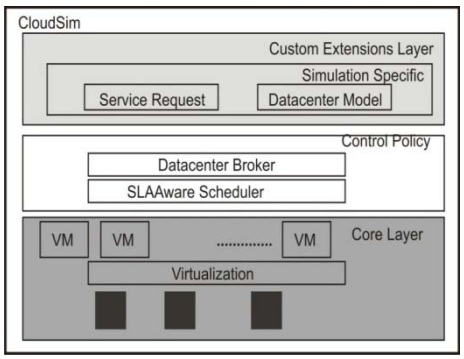

Figure 4 : Cloud Sim Extension Architecture

V. Evaluation

The evaluation of the scheduling heuristic demonstrates the resource utilization by the scheduler. It further shows the higher application performance obtainable while compared to an arbitrary task scheduler. The evaluations presented here are realized using the Clouds simulation tool [16].

Here, we begin with the experimental setup and configuration descriptions and basic experimental configurations. The experimental test bed is setup as described in Figure - 5. It demonstrates the processes of placing service request by customers and how our proposed scheduler deploys the service on appropriate Cloud resources.

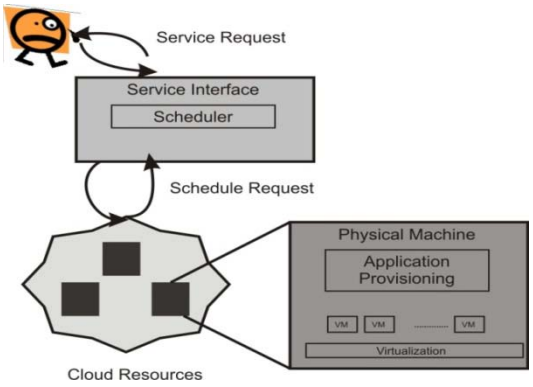

Figure 5 : Scheduling Evaluation Test bed

The Cloud resources comprise physical and virtual machines. Table - 1 shows the resource capacities of the physical machines and the configuration parameters of the virtual machines. Based on the capacities of the physical machine resources and the sizes of the virtual machines, we can start several virtual machines on one physical host in the Clouds simulation engine. To achieve a reasonable application deployment scenario, we use two types of applications each with its own SLA terms to realize heterogeneous workloads. The first workload is extracted from a Web Application (WA) for an online shop and the second workload is a trace of High Performance Computing (HPC) application represented by an image rendering applications such as POV-Ray.

Table 1 : Cloud Environment Resource Setup

Machine Type = Physical Machine

OS	CPU Core	CPU Speed	Memory	Storage	Bandwidth
Linux	6	6000 MIPS	3.072 GB	30000 GB	3 Gbit/s

Machine Type = Virtual Machine

OS	CPU Core	CPU Speed	Memory	Storage	Bandwidth
Linux	1	1000 MIPS	512 MB	5000 GB	500 Mbit/s

Table - 2 presents the experimental SLA objective terms for the two application types. The web application generally requires less resource while executing and its performance is ensured by the specified SLA objectives. The HPC applications are resource intensive in execution and their performance are safeguarded by the specified SLA objectives.

Guaranteeing these SLA terms ensures the good performance of the application executions.

Table 2 : Heterogeneous Application SLA Objectives

Application Type	CPU Power	Memory	Storage	Bandwidth
Web	240 MIPS	130 MB	1000 GB	150 Mbit/s
HPC	500 MIPS	250 MB	2000 GB	240 Mbit/s

a) Deployment Efficiency and Resource Utilization

The evaluation of the efficiency of the proposed scheduler for deploying customer service requests and utilizing the available Cloud resources. The test essence of the on-demand resource provisioning feature, simulate a large data center made up of 60 physical machines and 370 virtual machines. The capabilities of the scheduler are evaluated into two groups:

i) Fixed resource
ii) On-demand resource

In the fixed resource group the on-demand resource provisioning feature is deactivated while in the on-demand resource group, it is activated. The essence of these two groups is to demonstrate the advantages of the on-demand resource provisioning feature. Each group runs three scenarios:

i. The first scenario handles the deployment of only web applications' service requests.
ii. The second scenario deals only with HPC applications.
iii. The third scenario deals with a mixture of web and HPC applications.

The three scenarios are intended to cover real world deployment situations in the sense that they handle applications from different categories, which exhibit different behaviors in terms of resource consumption. In the scenarios, the service requests are randomly generated and sent to the scheduler for scheduling and deployment. Next, we describe the achieved results in two groups.

i. *Fixed resource group*

The scheduler schedules and deploys the applications on the available running VM in the data center without the flexibility of starting new VMs when required. The results achieved by the three scenarios of this group are presented in Figure - 6.

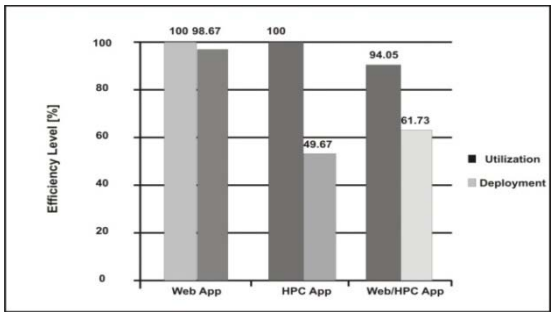

Figure 6 : Scheduling and Deploying With Fixed Resources

In Figure - 6, scenario 1 presents the results of the first evaluation scenario, which handles only web applications. The first bar shows the total resource utilization level achieved among the running VMs in the data center. The resource utilization is measured by checking the number of service applications. The scheduler achieved 100% resource utilization meaning that the resources on each VM were adequately utilized. The second bar shows the total deployment efficiency achieved by the scheduler. The deployment efficiency is calculated by counting the total number of deployed service applications in relation to the total number of requested services. In this scenario a total of 1480 service applications are deployed whereas a total of 1500 service requests were made. This gives a deployment efficiency of 98. 67%. About 20 service requests could not be provisioned due to lack of resources on the available VMs.

The results of the second evaluation scenario dealing with only HPC applications are presented as Scenario - 2 in Figure - 6. The first bar shows the resource utilization achieved by the scheduler, which is in this case 100%.. The second bar represents the deployment efficiency achieved, which is in this scenario 49.67%. The low deployment efficiency is caused by lack of available resources. The results of the third evaluation scenario are presented as Scenario 3 in Figure - 6. This scenario deals with a mixture of web and HPC applications' service requests. The scheduler achieved about 94.05% resource utilization in this scenario as shown by the first bar. The inability to achieve 100% resource utilization is caused by the heterogeneous nature of the workload whereby some

HPC applications cause some resource fragmentation leaving some resource fragments that are not usable by the scheduler. The second bar represents the deployment efficiency of this scenario, which is 61.73%. This is significantly better than the deployment efficiency achieved in the second scenario. This increase in deployment efficiency is attributed by the heterogeneous workload whereby the number of HPC applications' requests is smaller than in the second scenario.

ii. *On-demand resource group*

In this group, it is possible for the scheduler to flexibly start new VMs when necessary as far as there are available resources on the physical machines. This feature allows for higher service request deployment and better usage of the resources at the data center. The results obtained by the three evaluation scenarios of this group are depicted in Figure - 7. The first bar shows that the scheduler achieved 100% utilization in this case. The observation in this scenario as compared to the first scenario of the fixed group is the 100% deployment efficiency achieved, which is shown by the second bar. The scheduler made advantage of the flexible on-demand resource provisioning feature to start extra four virtual machines to fully deploy the whole service requests.

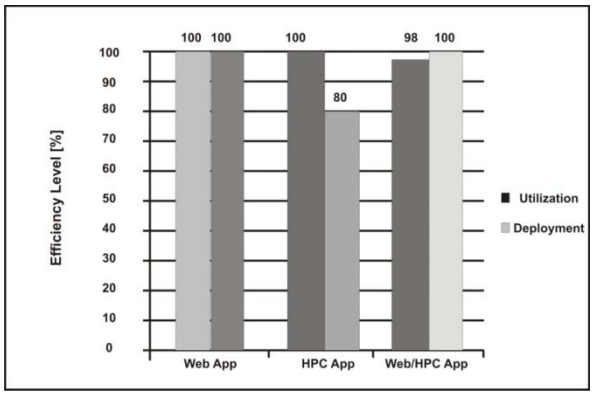

Figure 7 : Scheduling and Deploying with On-demand Resource Provisioning Feature

The second evaluation scenario results are presented as Scenario 2 in Figure - 7. This scenario deals with only HPC applications. The scheduler achieved 100% resource utilization in scheduling and deploying the HPC applications as depicted by the first bar. That means the available resources are fully utilized. Although the resources were fully utilized, the scheduler could only achieve 80% deployment efficiency. This is better result than 49.67% achieved by the equivalent scenario in the fixed group. The scheduler created extra 229 VMs for the applications deployments thereby reaching the limits of the physical machines and could not achieve 100% deployment efficiency due to ultimate lack of resources in the data center. This problem could be addressed with Cloud federation paradigm. Scenario 3 in Figure -7 depicts the results of the third evaluation scenario dealing with a mixture of web and HPC applications. The scheduler achieved 98% resource utilization due to resource fragmentations caused by the heterogeneous workload and resource over provisioning.

The last two VMs started on-demand were under-utilized. 100% deployment efficiency was achieved in this scenario by starting 215 VMs on-demand. Comparing the results achieved by the former group scenarios (Figure - 6) against those of the later group (Figure - 7), it can be clearly seen that the later group obtained much better resource utilization rates and deployment efficiencies. This demonstrates the effectiveness and relevance of our proposed scheduling approach in a Cloud environment.

b) Application Performance Comparison

The performance of applications being provisioned in the Cloud simulation test bed but application performance is evaluated in two aspects using the scenarios of the previous section:

i) Response time for the web applications
ii) Completion time for the HPC applications

The result achieved is compared by the proposed scheduler with that achieved by an arbitrary task scheduler. Table 3 presents the applications performance results. The results show the average response time and completion time of the applications while deployed by the two schedulers. It can be clearly seen that our proposed scheduler is two times better than the task scheduler. The good performance of our scheduler is attributed to the fact that it considers multiple performance objectives before deciding on which resource to deploy an application thereby finding the optimal resource combination for the application best performance, whereas the task scheduler considers mainly single objectives in its deployment, which cannot provide the optimal resources for the application best performance. Note that in Table 3 the on-demand resource provisioning feature applies only to our proposed scheduler.

Without On-demand Resource Provisioning Feature				
SLA-aware Scheduler			Traditional Task Scheduler	
Scenario	Response Time	Completion Time	Response Time	Completion Time
1	8 Sec	-	20 Sec	-
2	-	10 Sec	-	22 Sec
3	10 Sec	14 Sec	25 Sec	30 Sec
With On-demand Resource Provisioning Feature				
Scenario	Response Time	Completion Time	Response Time	Completion Time
1	5 Sec	-	15 Sec	-
2	-	7 Sec	-	18 Sec
3	8 Sec	10 Sec	19 Sec	24 Sec

Table 3 : Scheduler Comparison

VI. Conclusion

Scheduling and deployment strategies are means of achieving resource provisioning in Cloud environments. A further contribution of this thesis is the development of a novel scheduling heuristic considering multiple SLA objectives in deploying applications in Cloud environments. The heuristic includes load-balancing mechanism for efficient distribution of the applications' execution among the Cloud resources. A flexible on-demand resource usage feature included in the heuristic for automatically starting new VMs when non-appropriate VM is available for the application deployments is presented. The design of the heuristic and its implementations with proposed scheduling heuristic using the Clouds simulation tool is discussed.

In order to manage the deployment of multiple applications on a single virtual machine, a proposed application monitoring architecture (CASViD), which monitors and detects SLA violations at the application layer in Cloud environments. The evaluated architecture on a real Cloud test bed using three types of image rendering application workloads with heterogeneous behaviors necessary to investigate different application provisioning scenarios and to automatically determine the optimal measurement intervals to monitor the application provisioning. By experiments, the proposed architecture is efficient in monitoring and detecting individual application SLA violation situations. Further one can automatically find the optimal measurement intervals by sampling different ones and checking their net utility values. With the realization of CASViD, the capabilities of monitoring and detecting SLA violations of single customer applications being provisioned in a shared host, in addition to the previous resource monitoring techniques, a holistic monitoring model capable of monitoring at different layers in Clouds is provided.

References Références Referencias

1. G. Boss, P. Malladi, D. Quan, L. Legregni, and H. Hall, "IBM Cloud Computing (White Paper) ", Technical Report, http://download. boulder.ibm. com/ibmdl/pub/software/dw/wes/hipods/Cloud_co mputing_wp_final_8Oct.p df, accessed on 1st Nov. 2010.
2. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility", *Future Generation Computer Systems,* vol. 25, no. 6, pp. 599-616, 2009.
3. R.-S. Chang, J.-S. Chang, and P.-S. Lin, "An Ant Algorithm for Balanced Job Scheduling in Grids", *Future Generation Computer Systems,* vol. 25, no. 1, pp. 20-27, 2009.
4. J. Chen and Y. Yang, "Activity Completion Duration Based Checkpoint Selection for Dynamic Verification of Temporal Constraints in Grid Workflow Systems", *International Journal of High Performance and Computing Applications,* vol. 22, no. 3, pp. 319-329, 2008.
5. E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e- Science: An Overview of Workflow System Features and Capabilities", *Future Generation Computer Systems,* vol. 25, no. 6, pp. 528-540, 2008.
6. I. Foster, Z. Yong, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", *Proc. 2008 Grid Computing Environments Workshop (GCE08)*, pp. 1-10, Austin, Texas, USA, Nov. 2008.
7. X. Liu, D. Yuan, G. Zhang, J. Chen, and Y. Yang, "SwinDeW-C: A Peer-to- Peer Based Cloud Workflow System", *Handbook of Cloud Computing,*

B. Furth and A. Escalante (Eds), pp. 309-332, Springer, 2010.
8. I. J. Taylor, E. Edelman, D. B. Gannon, and M. Shields, *Workflows for e- Science: Scientific Workflows for Grids*: Springer, 2007.
9. L. Wang, W. Jie, and J. Chen, *Grid Computing: Infrastructure, Service, and Applications:* CRC Press, Taylor & Francis Group, 2009.
10. J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", *Journal of Grid Computing,* no. 3, pp. 171-200, 2005.
11. J. Yu and R. Buyya, "Workflow Scheduling Algorithms for Grid Computing", Met heuristics for Scheduling in Distributed Computing Environments, F. Xhafa and A. Abraham (Eds.), Springer, 2008.
12. Amazon. Amazon elastic computing cloud. http://aws.amazon.com/ec2/ (Last Access: April 3, 2012).
13. Ivona Brandic, Dejan Music, and Schahram Dustdar. Vieslaf framework: Facilitating negotiations in clouds by applying service mediation and negotiation bootstrapping. Scalable Computing: Practice and Experiences (SCPE), Special Issue of Scalable Computing on Grid Applications and Middleware & Large Scale Computations in Grids, 11(2):189– 204, June 2010.
14. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6):599 616, 2009.
15. Rodrigo N. Calheiros, Rajkumar Buyya, and Cesar A. F. De Rose. A heuristic for mapping virtual machines and links in emulation test beds. In International Conference on Parallel Processing, 2009. ICPP '09. Pages 518 –525, September 2009.
16. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya. Clouds: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41:23–50, January 2011.
17. Henri Casanova, Graziano Obertelli, Francine Berman, and Richard Wolski. The apples parameter sweep template: User-level middleware for the grid. Scientific Programming, 8(3):111–126, August 2000.
18. Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, and Ivan Breskovic. Sla aware application deployment and resource allocation in clouds. In 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops (COMPSACW), pages 298 –303, July 2011.
19. Vincent C. Emeakaroha, Ivona Brandic, Michael Maurer, and Schahram Dustdar. Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In 2010 International Conference on High Performance Computing and Simulation (HPCS), pages 48 –54, July 2010.
20. Google. Google appengine. https://developers google. Com/appengine/ (Last Access: April 3, 2012).
21. PeerHasselmeyer and Nicod'Heureuse. Towards holistic multi-tenant monitoring for virtual data centers. In Network Operations and Management Symposium Workshops, pages 350 –356, 2010.
22. Michael Maurer, Ivona Brandic, Vincent C. Emeakaroha, and Schahram Dustdar. Towards knowledge management in self-adaptable clouds. In Proceedings of the 2010 6th World Congress on Services, SERVICES '10, pages 527–534, 2010.
23. Radu Prodan and Simon Ostermann. A survey and taxonomy of infrastructure as a service and web hosting cloud providers. In 10th IEEE/ACM International Conference on Grid Computing, 2009, pages 17 –25, October 2009.
24. Sales Force. Servicecloud. http://www.salesforce. com/ (Last Access: April 3, 2012).

Global Journals Inc. (US) Guidelines Handbook 2013

www.GlobalJournals.org