DYNAMIC SPLITTING OF DEADLINE CONSTRAINT JOBS FOR PARALLEL PROCESSING IN CLOUD ENVIRONMENT

[1] MS. DHARMISHTHA J GHOGHRA, [2] MS. RICHA SINHA.

[1]M.E. [Information Technology] Student
[2] Asst.Professor [Department Of Information Technology]
KITRC, Kalol, Gujarat

shreeji2011@gmail.com; richa.872005@gmail.com

***ABSTRACT*:** *Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network typically the Internet. The infrastructure of cloud computing environment is usually composed of hundreds or even thousands of server nodes. Nowadays applications ranging from simple web site to large scale parallel e-commerce site are deployed to Cloud infrastructure, mainly due to As-You-Go and competitive billing rates by various cloud service providers, easy or no maintenance of IT resources and no initial investment required in IT infrastructure.*

An increasing number of high performance computing parallel applications leverage the power of the Cloud for parallel processing. How to schedule the parallel applications to improve the quality of service is the key to the successful host of parallel applications in the Cloud. A job submitted to Cloud can be divided in number of tasks to run them parallel. If all divided tasks are independent of each other then they can be easily run in parallel, but tasks have dependencies amongst themselves then cannot be easily made to run parallel on Cloud.

Most of the existing work in parallel job provisioning and scheduling does the static splitting of job into tasks, while some work provides dynamic provisioning but they assume the tasks of independent of each other. This may result in missing deadline for parallel jobs.

In this paper I proposed dynamic application provisioning algorithm to ensure job completion in deadlines specified in cost effective way for parallel tasks. Solution identified in research work replaces static job splitting by dynamic job splitting to ensure meeting the deadlines.

Keywords— CloudSim, Simulation, Provisioning, Parallel Computing, Independent and Dependent tasks in cloud.

I. INTRODUCTION

Nowadays applications ranging from simple web site to large scale e-commerce applications are being deployed to cloud infrastructure. Mainly due to as you go or competitive billing rates applied by various cloud service providers and easy maintenance in terms of physical hardware infrastructure and no initial investment in infrastructure. But still before going to cloud one need to analyse his application in terms of complexity, SLA (Service Level Agreement) requirements like response time and other performance aspects. Applications targeting mid or large range of audience are generally developed in 3-tier Architecture (Web Tier, Application Tier and Database Tier) and require parallelism of task at a time. While on other hand simple web application targeting relatively small amount of audience do not required parallelism at all.

In this research I tried to explain first, how one can use CloudSim (simulator for cloud) for the comparison of the communicating and non-communicating tasks required for Application to be deployed in cloud. The performance analysis is done on the bases of data transfer, time taken and user debts for both the process. Secondly, this explains current static job splitting mechanism employed by Cloudsim (cloud simulator) and finally explains how dynamic splitting can be implemented in CloudSim for better performance especially for parallel jobs.

II. CLOUD COMPUTING SERVICE MODELS

Cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services)

that can be rapidly provisioned and released with minimal management effort or service provider interaction. It is a style of computing where massively scalable IT-related capabilities are provided "as a service" across the internet to multiple external customers [5]. This term effectively reflects the different facets of the Cloud Computing paradigm which can be found at different infrastructure levels.

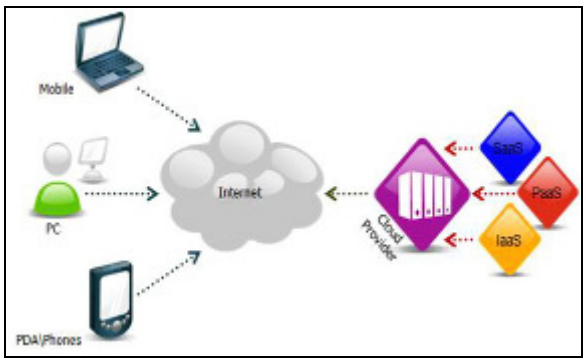

Figure.1. Cloud Computing Service Models

From figure.1, it is possible to identify three Cloud Service Models, namely the Software as a Service (SaaS) model, the Platform as a Service (PaaS) model and the Infrastructure as a Service (IaaS) model.

A. *SOFTWARE AS A SERVICE MODEL*

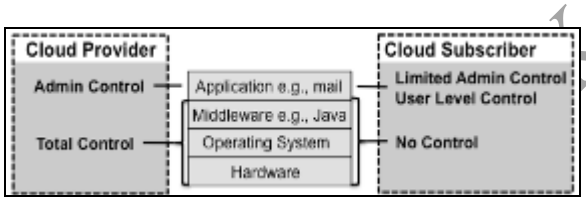

Figure.2. Level of control in SaaS Model

Figure.2 illustrates the relative levels of control between the provider and the subscriber. Here the consumer is free of any worries and hassles related to the service. The Service Provider has very high administrative control on the application and is responsible for update, deployment, maintenance and security. The provider exercises final authority over the application. For example, Gmail [5] is a SaaS [9] where Google is the provider and we are consumers. We have very limited administrative and user level control over it, although there is a limited range of actions, such as enabling priority inbox, signatures, undo send mail, etc, that the consumer can initiate through settings.

B. *PLATFORM AS A SERVICE MODEL*

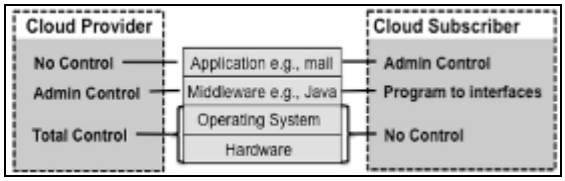

Figure.3. Level of control in PaaS Model

Figure.3 shows PaaS [11] Component Stack and Scope of Control. PaaS is a platform where software can be developed, tested and deployed. It means the entire life cycle of software can be operated on a PaaS. This service model is dedicated to application developers, testers, deployers and administrators. A PaaS typically includes the development environment, programming languages, compilers, testing tools and deployment mechanism.

C. *INFRASTRUCTURE AS A SERVICE MODEL*

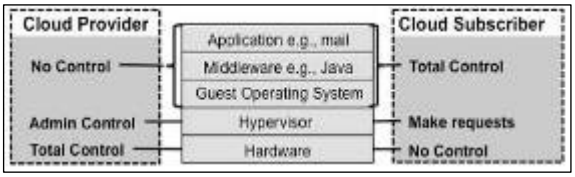

Figure.4. Level of control in IaaS Model

Figure.4 shows IaaS Component Stack and Scope of Control. IaaS is what the user should opt from virtual computers, cloud storage, network infrastructure components such as firewalls and configuration services. Usage fees are calculated per CPU hour, data GB stored per hour, network bandwidth consumed, network infrastructure used per hour, value added services used, e.g., monitoring, auto-scaling etc. The most popular facebook games, *Farmville* [8] and *Mafia Wars,* has more than 230 million monthly users run more than 12000 servers on Amazon AWS. When they launch a new game, they start with a few servers and then ramp up their capacity in real time. Most important among the lot are SaaS and PaaS players who are hosted with IaaS providers.

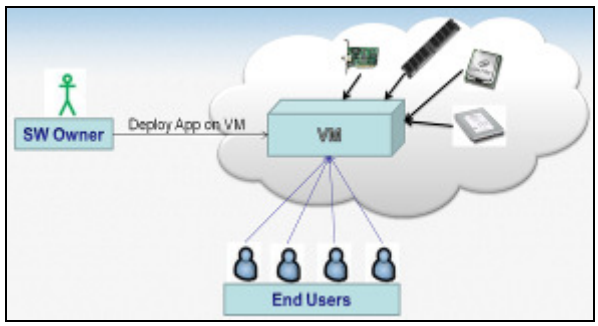

Figure.5. IaaS Model

III. PROVISIONING

Definition: "Allocation of a cloud provider's resources to a SW Owner"

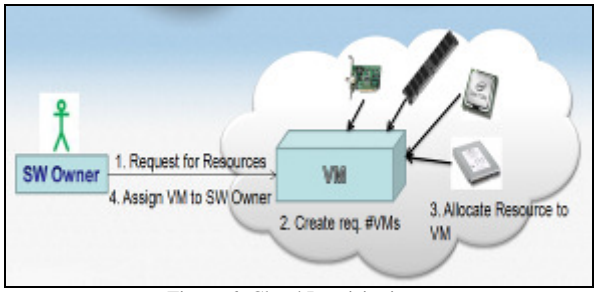

Figure.6. Cloud Provisioning

Cloud provisioning is the allocation of a cloud provider's resources to a customer. When a cloud provider accepts a request from a customer, it must create the appropriate number of virtual machines (VMs) and allocate resources to support them. The process is conducted in several different ways: Advance provisioning, dynamic provisioning and user self-provisioning. In this context, the term provisioning simply means "to provide."

The process is conducted in several different ways:

- With Advance provisioning, the customer contracts with the provider for services and the provider prepares the appropriate resources in advance of start of service. The customer is charged a flat fee or is billed on a monthly basis.
- With dynamic provisioning, the provider allocates more resources as they are needed and removes them when they are not. The customer is billed on a pay-per-use basis. When dynamic provisioning is used to create a hybrid cloud, it is sometimes referred to as cloud bursting.

With user Self-provisioning (also known as cloud self-service), the customer purchases resources from the cloud provider through a web form, creating a customer account and paying for resources with a credit card. The provider's resources are available for customer use within hours, if not minutes.

A. APPLICATION PROVISIONING

Cloud computing provides application owners with instant access to infrastructure, however building out applications still requires administrators to individually install and configure application components on each virtual machine to plug into the application architecture.

Application Provisioning is a Service to deploy of deploying specialized applications within Virtual Machines and mapping end user request to application instances.

B. VIRTUAL MACHINE PROVISIONING

Definition: "Deployment of specialized applications within VMs".

It is the main point of contact in the system that receives incoming requests and creates VMs. The Admission Control module is responsible for admitting service requests for allocation and maintains a buffer of requests which cannot be serviced immediately on arrival.

Virtual Machine provisioning, this involves instantiation of one or more Virtual Machines (VMs) that match the specific hardware characteristics and software requirements of an application.

C. RESOURCE PROVISIONING

Resource Provisioning is the mapping and scheduling of VMs on physical Cloud servers within a cloud. It allocates Virtual Machines to instances of an application and follows two phases for Resource Provisioning.

IV. CLOUDSIM ARCHITECTURE

The CloudSim[4] simulator is currently the most sophisticated discrete event simulator for Clouds. It has many features which made us choose it for building our simulation environment on top of it. Figure 7 shows components of the CloudSim Architecture [4]. In this section, we outline the functionality of different layers of CloudSim.

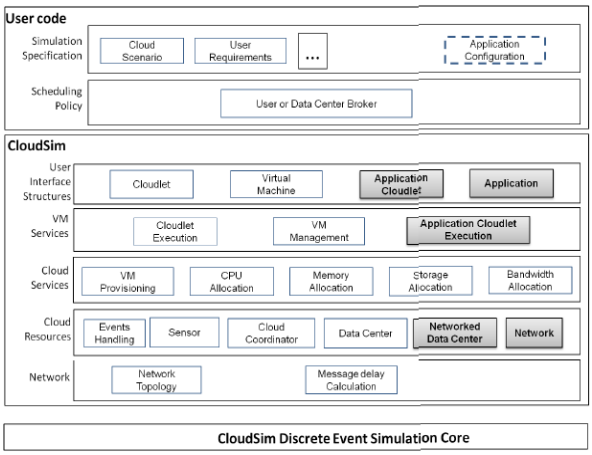

Figure.7. Layered Architecture of CloudSim with Network CloudSim elements

The bottommost layer of the CloudSim architecture handles the interaction between CloudSim entities and components. All components in CloudSim communicate through message passing operations. The second layer consists of several sub-layers that model the core elements of Cloud computing. The bottommost sub layers model datacenter, Cloud coordinator and network topology

between different datacenters. These components help in designing IaaS infrastructure. The VM and Cloud Services provide the functionality to design resource (Virtual Machine (VM)) management and application scheduling algorithms. The layers above help users to define their own simulation scenarios and configurations for validating their algorithms. In this paper, we incorporate a generalized application model and components to design arbitrary network topologies within datacenters.

IV. APPLICATION MODEL

Figure.8 shows the Cloudsim components for application programming.

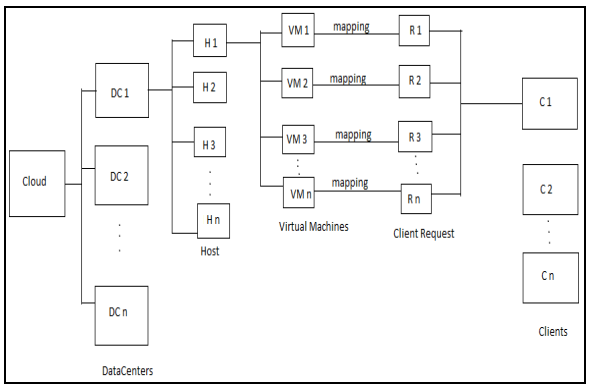

Figure.8. CloudSim: Application model/components

To model the application itself, a basic and general structure (i.e. a Java class), called AppCloudlet is defined. Each AppCloudlet object consists of several communicating elements (NetworkCloudlet). Each element runs in a single virtual machine and consists of communicating and computing stages. Each computation stage can be defined either by the number of MIPS or seconds involved in it. The communications are characterized by the amount of transferred data.

V. SIMULATION OF PARALLEL APPLICATION

There are three main actors (or Entities) in the CloudSim: Switch, NetworkDatacenter, and NetworkDatacenterBroker [4].

A. SWITCH

It represents a network entity which can be configured as a router or switch. It can model delays in forwarding any data to either host or another switch based on where the data belongs.

B. NETWORK DATACENTER

It represents a network entity which can be used to configured datacenter for the network management.

C. NETWORK DATACENTER BROKER

It represents a network entity which can be used to configured broker policies for the network. This entity basically represents the provisioning part of resources, usage, VM and other parameters.

The class diagram for NetworkCloudsim[4] is shown in figure 9.

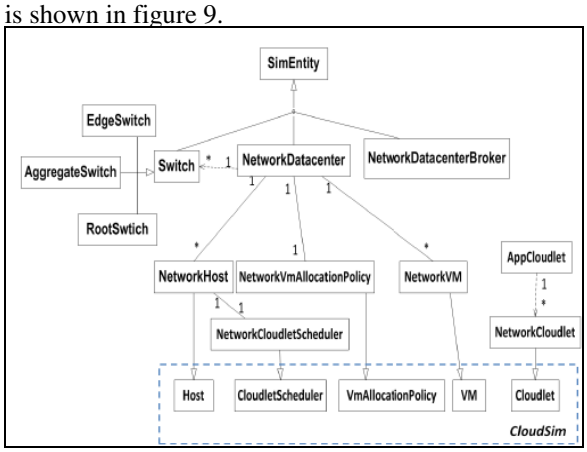

Figure.9. Class Diagram of NetworkCloudSim

In case of NetworkCloudsim, a job is divided into tasks. This job parallelism is done under two categories:

VI. JOB HIERARCHY

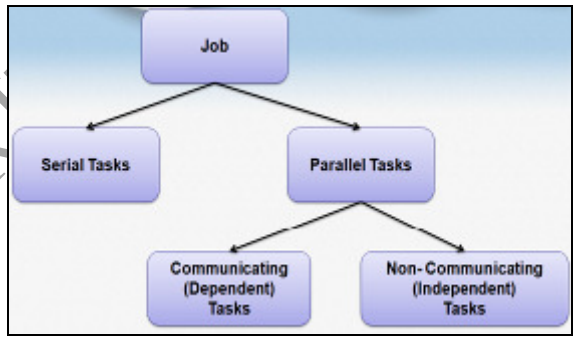

Figure.10. Job Hierarchy

Figure.10. shows there are basically two type of jobs in Cloud environment either it can be Serial Job or it can be Parallel job. Serial job is one which all tasks needs to be run in sequence, in other word we cannot run tasks of serial job in parallel.Parallel job is one which all or some tasks can be parallelized. Parallel jobs' tasks can be further divided in communicating or non communicating tasks.

A. SERIAL VS. PARALLEL TASKS

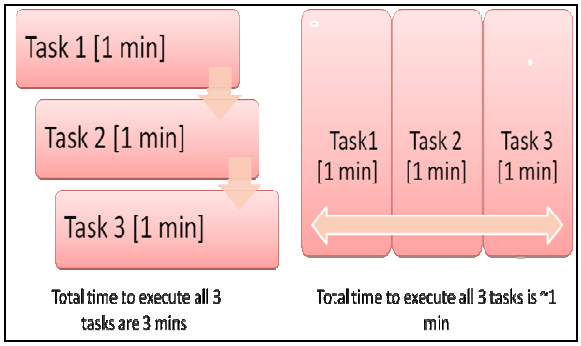

Figure.11. Serial Vs Parallel Tasks in cloud

Figure.11. shows if it is to be assumed that a given job can be divided in three independent tasks and every task takes 1 minute to execute. If all tasks run in sequence i.e. serial then total time taken would be 1 + 1 + 1 = 3 minutes. While in case of parallel tasks, all tasks can be schedule to run in parallel independent of one another and hence they all can complete their execution within 1 minute.

VI. PROPOSED SYSTEM

A. *PROPOSED SYSTEM MODEL*

System and application models assumed jobs are composed of either independent or dependent tasks and they can also contain deadline and budget specification. The Resource Management System (RMS) deploys Hybrid Cloud resources to execute tasks and meet deadlines. Decisions are made by the RMS with the support of information about users, groups they belong to, and their access rights.

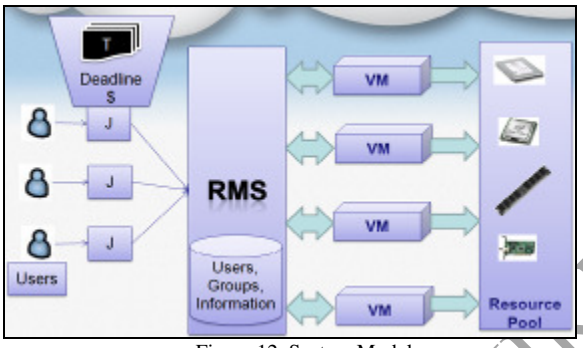

Figure.12. System Model

The system model assumed in this paper is depicted in above figure. The central component of the model is the Resource Management System (RMS) that manages scheduling and provisioning of resources. Resources manage by provisioner are like CPU, Memory, Network Bandwidth and Storage. Upon receiving request from user RMS will allocated enough number of resources to VMs and finally allocate those VMs to users.

The RMS is accessed by users who want to submit loosely-coupled distributed applications in the resources managed by the system. The user request (job) contains (i) description of each task that composes the job, including required estimated runtime (execution time); and (ii) optional QoS attributes in the form of deadline for job completion and budget to be spent to meet the deadline.

Tasks from different users compete for resources, and the RMS determines which tasks execute in a given moment and where. However, this has to be done without causing starvation to any job in the waiting queue (i.e., the RMS has to guarantee that each job will eventually complete). Furthermore, the organization can enforce policies about access rights of users and groups, which have to be taken into account by the RMS.

When the RMS detects that one or more jobs are risking missing their deadlines, provisioning policies are applied so that resources are acquired and deployed to speed up such jobs.

B. *Proposed System Structure*

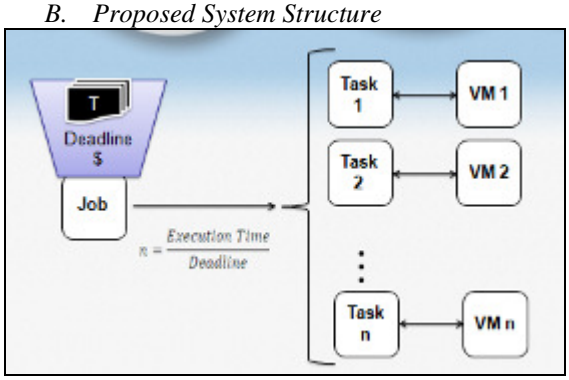

Figure 13 Proposed Structure

When user submits Job (J), he specifies following things:

1. No of tasks in Job (Nt) and respective execution time in seconds of task (Tie)

   Where Ti, Ti+1… Tn are tasks specified by User
2. Deadline (Dj) for the Job (in seconds)
3. Budget ($) constraints if any

User specified some task can be too large to fit into one VM or some tasks are too small that multiple such tasks can be run in single VM. If RMS schedule task according to what user says then either deadline be missed; Or ended up allocating more than required No of VMs to given job which ultimately cost to user. To overcome above said limitation I propose that RMS should dynamically split job into tasks according to following formula.

No of Logical Tasks (Nlt) = Sum of Execution Time for all tasks (∑ Ti) / Deadline (Dj)

Above thing can be better explained by following example. If User submit a Job (J) with deadline of 3 seconds, and he specifies that job can be divided into 6 tasks (T1 to T6) and every task can be executed in 1 sec.

So as per formula: Nlt = (1 + 1 + 1 + 1 + 1 + 1) /3 = 6/3 = 2

That mean though user said job can be divided in to 6 tasks, RMS will divide 6 tasks into 2 logical categories of tasks and schedule each logical category of task in to separate VM. So total 2 VM will required instead of 6.

C. *ALGORITHM for PROPOSED WORK*

Following algorithm will be used for future work.

INPUT:

1. No of tasks in Job (Nt)
2. Respective execution time in seconds of task (Tie)
   Where Ti, Ti+1… Tn are tasks specified by User
3. Deadline (Dj) for the Job (in seconds)
4. Budget ($) constraints if any

OUTPUT:
Allocation of No of VMs to tasks so job can be completed within deadline specified

STEPS:
Step 1:
Calculate the job execution time from task execution time given by user.
Job Execution Time (Tj) = Sum of Execution Time for all tasks (∑ Ti)

Where Ti, Ti+1… Tn are tasks specified by User

Step 2 :
Job Exec Time (Tj) = Sum of Exec time of all tasks (Σ Tie)

If (Job Execution Time (Tj) > Deadline (Dj))
- Split job into n logical tasks; n= Tj / Dj
- Create n virtual machines
- Assign n tasks to n virtual machines

Else
- Create 1 virtual machine
- Assign 1 job to 1 virtual machine

Return

VII. CONCLUSION AND FUTURE WORK

Conclusion can be derived that existing implementation of Cloudsim splits the job in static manner for the parallel jobs. The proposed work will apply dynamic splitting to deadline constraint jobs, so deadline cannot be missed.

In future, I am planning to implement this theory on CloudSim and check for the working parameters. As the research work for this area is comparatively low so various other parameters like CPU utilization, No of Virtual machines, Cost, Deadline, SLA etc. can be studied so dynamic task splitting can be more efficient

VIII. ACKNOWLEDGEMENT

I would also like thank my husband to Mr. Jadav Bheda for helping me through out with this research work.

IX. REFERENCES

[1] Lizhe Wang, Jie Tao, Kunze M, Castellanos A.C., Kramer D., Karl W., "Scientific Cloud Computing: Early Definition and Experience", High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference

[2] Turban E, King D, Lee J, Viehland D, "Chapter 19: Building E-Commerce Applications and Infrastructure", Electronic Commerce A Managerial Perspective (5th ed.), Prentice-Hall, pp.27, 2008

[3] Buyya R, Yeo C, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 2009; 25(6):599–616.

[4] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling andSimulation of Scalable Cloud Computing Environments and theCloudSim Toolkit: Challenges and Opportunities" Proc. of the 7th High Performance Computing and Simulation Conference (HPCS09), IEEE Computer

[5] Thepparat T., Harnprasarnkit A., Thippayawong D., Boonjing V., Chanvarasuth P., "A Virtualization Approach to Auto-Scaling Problem", Eighth International Conference on Information Technology: New Generations (ITNG), 2011

[6] S.Garg and R. Buyya,, "NetworkCloudSim:Modeling Parallel Applications in Cloud Simulations, IEEE Computer Society, 2011

[7] Buyya R, Yeo C, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 2009; 25(6):599–616.

[8] Google App Engine, http://code.google.com/intl/en/appengine/

[9] Daniel Nurmi , Rich Wolski , Chris Grzegorczyk , Graziano Obertelli , Sunil Soman , Lamia Youseff , Dmitrii Zagorodnov, "Eucalyptus : A technical report on an elastic utility computing architecture linking your programs to useful systems (2008)", published in UCSB TECHNICAL REPORT

[10] Feng Liu, Weiping Guo, Zhi Qiang Zhao,Wu Chou, "SaaS Integration for Software Cloud", published in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference, Pages 402-409