Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 4 (2023)

Compelling method for managing Further foster System Execution and Cost Evaluation for Cloud

[1] Pavan Kumar Panakanti, [2]Dr. James Gladson Maria Britto, [3]K. Sandhya Vani, [4]Sheri Ramchandra Reddy, [5]M. Shiva Priya, [6]Dr. Sunil Tekale, [7]Dr. Shyamsunder P. Kosbatwar

[1] Associate Professor, CMR Institute of Technology. Hyderabad
[2]Professor, CSE, Malla Reddy College of Engineering, Hyderabad
[3]Asst. Professor, Malla Reddy College of Engineering, Hyderabad
[4]Assistant Professor, CMR Technical Campus, Computer Science and Engineering (AI&ML), Hyderabad
[5]Asst. Professor, Malla Reddy College of Engineering, Hyderabad
[6]Professor, Dept. of CSE, Malla Reddy College of Engineering, Hyderabad
[7]Associate Professor, Dept. of CSE, Dnyanshree Institute of Engineering & Technology, Satara

E-mail: [1]pavan.panakanti@cmritonline.ac.in, [2]panakanti@yahoo.com, gmbritto@gmail.com, [3] sandhyaganguri@gmail com, [4]ramchandra.ram123@gmail.com, [5]shiva6priya@gmail.com, [6]sunil.tekale2010@gmail.com, [7]shyamsunder.kosbatwar@dnyanshree.edu.in

Abstract: Cloud Computing is assuming a relevant role in the world of web applications and web services. On the one hand, cloud technologies allow realizing dynamic system which are able to adapt their performance to workload fluctuations. On the other hand, these technologies allow eliminating the burden related to the purchase of the infrastructure, allowing more flexible pricing models based on the actual resource utilization. Last, but not least, is the possibility to completely delegate to the Cloud Provider intensive tasks as the management and the maintenance of the cloud infrastructure. Moreover, the usage of cloud systems can lead to relevant issues, which mainly derive from the lack of technology standards and from the intrinsic characteristics of such geographically distributed systems. For example, we can mention the lock-in effect related to the portability of cloud applications, the problem of data location and data security, the lack of interoperability between different cloud systems, the problem of performance and cost estimation.

This paper is focused on the problem of performance and cost estimation of cloud system at IaaS (Infrastructure-as-a-Service) and PaaS (Platform-asa- Service) level, which is crucial for service providers and cloud end users. These latter need valid comparison metrics, in order to choose whether or not to use cloud technologies and, above all, on which Cloud Provider they can rely. The derivation and the analysis of these metrics are not straightforward tasks, since cloud systems are geographically distributed, dynamic and therefore subject to high variability.

Keywords: Cloud Services, Performance, Cost evolution, Dynamic responsive

1. Introduction

In a world of fast changes, dynamic systems are required to provide cheap, scalable and responsive services and applications. The Cloud Computing is a possible solution, a possible answer to these requests. Cloud systems are assuming more and more importance for service providers due to their cheapness and dynamicity with respect to the classical systems. Nowadays there are many applications and services which require high scalability, so that for service providers the in-house management of the needed resources is not convenient. In this scenario, cloud systems can provide the required resources with an on-demand, self-service mechanism, applying the pay-peruse paradigm. The approach we have used is different from the ones which are generally adopted by other several cloud related projects. In the next we will discuss about such projects and approaches, which are generally based on multi-cloud libraries, so they try to face cloud issues like interoperability, compatibility and portability working at API level.

The spread of cloud systems has unearthed the other side of the medal: if we use these systems, we have to take into account problems in terms of quality of service, service level agreements, security, and

compatibility, interoperability, cost and performance estimation and so on. For these reasons, many cloud related projects have been developed around the concept of Multi- Cloud, which is intended to solve most of the aforementioned issues, especially compatibility and interoperability. We will discuss about Multi-Cloud and related projects in this paper.

2. Related Data

Nowadays there are several definitions of Cloud Computing, but the one given by the National Institute of Standard Technology (NIST) looks the most accurate [1]: Cloud computing is a model for enabling ubiquitous, 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.

This definition highlights the basic properties of Cloud Computing: *Ubiquity:* the user can totally ignore the location of the hardware infrastructure hosting the required service and can use the service everywhere and every time through his client application. *Convenience:* the consumer can use a service exploiting remote physical resources, without necessity of buying/acquiring those resources. He just uses the resources provided by the provider and pays for them with a pay-per-use mechanism. *On-demand activation:* a service consumes resources only when is explicitly activated by the user, otherwise it is considered inactive and the resources needed for its execution can be used for other purposes.

The NIST definition also specifies five essential characteristics of Cloud Computing: *On-demand self-service:* a consumer can use the resources without any interaction with the service provider, with an on-demand policy. *Broad network access:* resources are available through the Internet and can be accessed through mechanisms that promote their use by simple (thin) or complex (thick) clients. *Resource pooling:* physical and virtual resources are pooled to serve many users and are dynamically assigned with respect to the users' needs and requirements. *Rapid elasticity:* resources can be rapidly and elastically provisioned; the consumer often perceives "unlimited" resources that can be purchased in a very short time. *Measured service:* resources use is always automatically controlled and optimized by monitoring mechanisms at a level of abstraction appropriate to the type of service (e.g., CPU activity time for processing services and so on).

We can distinguish three main service models in the world of Cloud Computing:

- Software-as-a-Service (SaaS): the consumer uses a provider's application running on a cloud infrastructure. The user can manage only limited user-specific application settings and cannot control the underlying infrastructure.
- Platform-as-a-Service (Paas): the consumer can deploy on the cloud infrastructure owned or acquired applications created using programming languages and tools supported by the provider. The user can control the application and the deployment settings, but cannot manage the underlying infrastructure, or the allocated resources.
- Infrastructure-as-a-Service (IaaS): the consumer can deploy and execute any kind of software on the cloud infrastructure. The user cannot control the underlying infrastructure, but has control over the deployed applications, Operating System, storage, and some network components (e.g., firewalls) and it is responsible of the management of the resources.

Finally, we can distinguish different deployment models: *Public Cloud*: the cloud infrastructure is made available to the general public and is owned by a private organization selling cloud services. *Community Cloud:* the cloud infrastructure is shared among several organizations and supports a specific community with shared concerns. It can be managed by the organizations or a third party and may exist on-premise or off-premise. *Private Cloud:* the cloud infrastructure can be accessed only within the organization and can be managed by the organization itself or a third party and may exists on-premise or off-premise. *Hybrid Cloud:* the cloud infrastructure is composed by different autonomous clouds connected together with a standard or proprietary technology that enables data and application portability.

At the present time, the main problem in the world of cloud computing is the absence of common standards. This leads on the one hand to a lack of uniformity and on the other to a large amount of proprietary incompatible solutions. We can identify several common weaknesses among the actually available cloud services: Often there's no way to manage the Service Level Agreements (SLAs) and to monitor the Quality of

Service (QoS). Lack of infrastructural uniformity and interoperability among different cloud providers. Lack of software uniformity (different platforms and APIs) among different cloud providers. Security issues on access and use of cloud services.

3. Proposed Schema

The last research area is that regarding the so-called *Sky-Computing*, a particular architecture lying above the cloud computing architecture. It is a sort of middleware that control and manage an environment of clouds, offering variable storage and computing capabilities [5, 6]. In other words, it consists in the interconnection and provisioning of cloud services from

Multiple domains. So it combines the infrastructural interoperability, typical of an Inter-Cloud architecture, and the uniformity at PaaS level, typical of a Multi-Cloud infrastructure. The Sky-Computing aims to maximize the interoperability among the clouds and to achieve the maximum elasticity for the cloud-based applications and services. Finally, within a Sky-Computing architecture it is possible to perform SLA management and QoS monitoring, so such architecture is particularly suitable to realize High Performance Computing (HPC) systems.

The Figure 1 shows the state of the art in the previously described areas and highlights the classification of the open-source projects that will be analyzed in the following. There are several specific projects for the SLA management and QoS monitoring, but in general these problems are faced in most of the projects in the fields of Inter-Cloud and Multi-Cloud.

Performance and cost evaluation of cloud systems is a key point in choosing the best cloud solution when we have to deploy a certain cloud application. In this we will present general and specific cloud meta-models which are used to represent the relevant services and features of cloud systems.

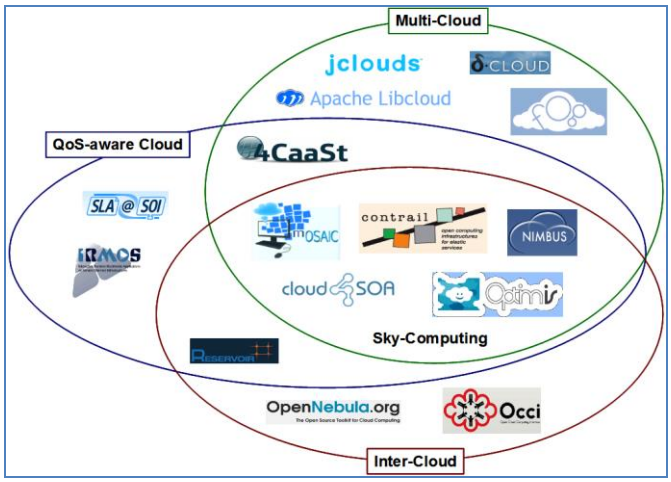

Fig 1: Cloud World

We will discuss about the clouds and meta-models and about the objectives we want to achieve using these meta-models. In this context, In order to evaluate performance and cost of cloud applications, a general model representing a cloud environment is needed. Using different levels of abstraction it is possible to model cloud systems and to evaluate performance indices and costs of cloud applications running on them with different granularity levels.

The goal of this paper is to extend the Proposed Scheme implementing a new tool System Performance and Cost Evaluation for Cloud, which will be described in the next. Cloud applications performance and costs will be derived proposing suitable cloud meta-models and generating a mapping between these meta-models and the Proposed Component Model (PCM). With this mapping it is possible to use the features of Proposed to evaluate performance and cost of cloud applications.

An important parameter that must be considered when comparing different cloud providers or different configurations is the cost of the selected resources. Cloud resources are always characterized by certain costs which depend on the quality/amount of the virtual hardware resources they provide, so an attribute *cost* is

needed. It is important to notice that there could be a dependency between the cost of a cloud resource and the time of the day. This is the case, for example, of the Amazon EC2 Spot Instances which have time-variant pricing models. In order to represent this dependency we cannot consider a single attribute *cost* in our model, but we should introduce a *cost profile* which characterizes the cost of a given resource in a given time period.

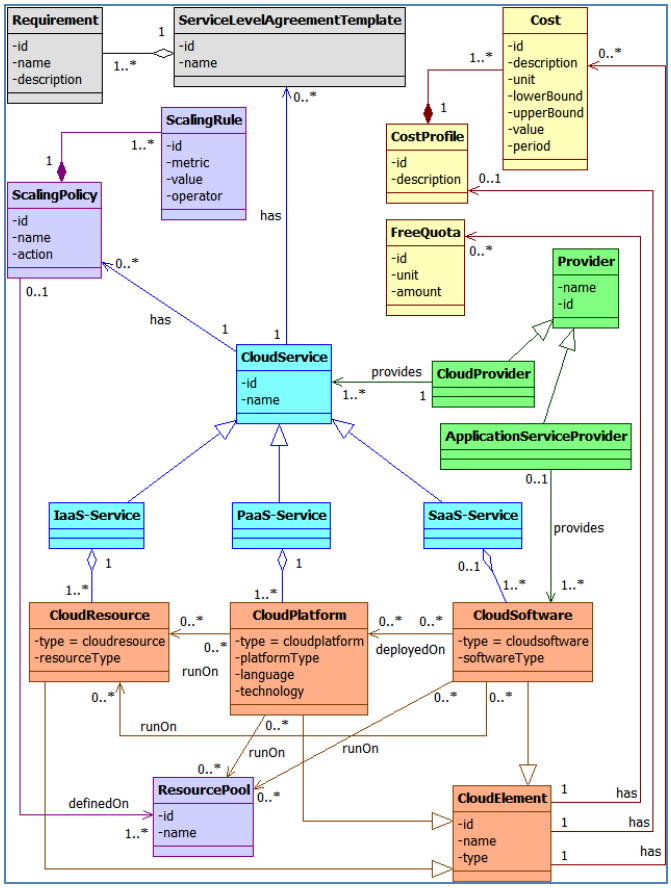

Fig 2: Proposed Architecture

4. Implementation - Performance and Cost Metrics

A Cloud Platform is a software framework exposing a defined Application Programming Interface (API) that can be used to develop custom applications and services. The platform also provides an execution environment for such custom applications and services. A Cloud Platform always runs on at least a Cloud Resource. Frontend, Middleware, Backend and Database platforms are three possible specializations of a Cloud Platform. Frontend platforms can host frontend services, which are directly exposed to the end users and are supposed to interact with them providing data to backend services. Backend platforms can host backend services, which are hidden to the end users and are supposed to process data coming from frontend services eventually providing them some intermediate results. Middleware platforms can host services like message queues or task queues which are used to decouple Frontend instances from Backend instances. A Database platform is able to store structured or semi-structured data and can be classified into Relational DB or NoSQL DB. A Relational DB is a Database platform

Based on the relational model, so database entries are organized into tables and can be accessed and manipulated through relational query languages. A NoSQL DB is a Database platform based on a distributed architecture, into which data are not required to have a relational structure. Furthermore, these databases use query languages different form SQL and they cannot guarantee all the ACID properties (Atomicity, Consistency, Isolation, Durability). Databases are considered as cloud platforms because they provide interfaces to access structured or semi-structured data and can be configured by the user, but it is not possible to control their underlying infrastructure (IaaS level).

Even if we can use a one-to-one mapping for concepts like CPU and Storage, we have to take into account that in Proposed these resources are expressed numerically in terms of processing rates. So an appropriate way of expressing the computational power and the storage processing power with simple numbers must be found. This operation is necessary due to the fact that at low level (i.e. the LQN layer) each resource is represented with a queuing network node, so we need to know which the processing rate of each resource is.

Expressing the CPU processing rate with a simple number is not a trivial task, because it depends on many different factors and usually it is empirically determined by benchmarking the CPU. Considering that the *processing Rate* is defined in terms of number of operations executed in one second, a convenient way to represent this parameter is to use the CPU frequency value. But if we consider Amazon instances, their processing power is expressed in terms of ECUs and the frequency value is not available. Furthermore, this kind of measure unit is not adopted by other cloud providers, which in most cases simply report the common CPU specifications such as the vendor and the model, the number of cores, the frequency. Since there exist a lot of public benchmark tables for all the common CPUs on the market, we could choose a benchmark score to represent the computational power of a given CPU. Figure 3 shows the CPIM UML lass Diagram derived from the *Resource Model*

The component specification is performed by specifying a Proposed Repository Model, and then the SD can connect and organize components defining the internal structure of the system within the Proposed System Model.

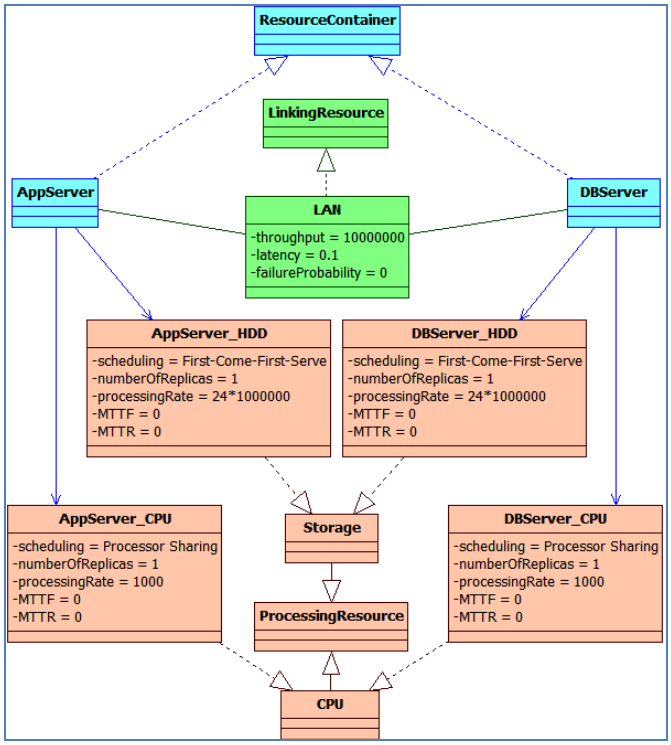

Fig 3: Resource Environment UML

At this point, the Software Designer specifies at a high level of abstraction which resources will host the system defining the Proposed Resource Model. In this step the SD is required to specify only the Resource Containers within the Resource Model, as a sort of black boxes, without defining their Processing Resources. The Resource Model specification will be completed at a later time by the Performance Engineer. The last step performed by the SD is to define the Proposed Allocation Model, specifying which components of the system are allocated on which resource containers.

For what concerns performance evaluation, the tool exploits the proposed features to run the 24 hours analysis. As anticipated, the tool gives the possibility to choose either the LQN Solver (LQNS) or the SimuCom simulator to run the analysis. The main difference between the two resolution methods is that LQNS is a very fast analytic solver, while SimuCom is a simulator, so it requires more time to give results. Furthermore, using

SimuCom it is possible to analyze the results hour by hour in a dedicated Eclipse perspective, with the possibility to show interesting graphs about response times, resource utilization and so on. Using LQNS, instead, it is possible to obtain human readable results, but then it is up to the Performance Engineer to elaborate them to obtain more interesting information.

5. Conclusions

Cloud Technologies are promising but performance and cost evaluation are challenging for service providers that decide to deploy their applications on cloud systems. This paper provides a model-driven approach to address this problem, following the Clouds vision. We have proposed a Cloud Provider Independent Model (CPIM) to represent general cloud systems focusing on those aspects which affect performance and costs. From this general representation we have derived examples of Cloud Provider Specific Models (CPSMs) related to Amazon Web Services, Microsoft Windows Azure, and Flexi scale. We have demonstrated that is possible to integrate these representations with the existing performance and cost evaluation tools extending their performance and cost analysis capabilities to cloud systems. In particular, we have extended the Proposed Framework, using parts of the Proposed Component Model as a Cloud Independent Model (CIM), and integrating within the tool suite our CPIM and CPSM models.

References

[1] Peter Mell, Timothy Grance, The NIST Definition of Cloud Computing. National Institute of Standard Technology, U.S. Department of Commerce. January 2011.
[2] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, Monique Morrow, Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability. Internet and Web Applications and Services, International Conference on, pp. 328-336, 2009 Fourth International Conference on Internet and Web Applications and Services, 2009.
[3] Sam Johnston, The Intercloud is a global cloud of clouds, 22 June 2009. http://samj.net/2009/06/ intercloud-is-global-cloud-of-clouds.html
[4] Michael Crandell, Josep M. Blanquer, How To Think Multi- Cloud. RightScale Webinar, 08/12/2010. http://www.slideshare.net/ rightscale/rightscale-webinar-how-to-think-multicloud
[5] Monteiro A., Pinto J.S., Teixeira C., Batista T.. Sky computing. Information Systems and Technologies (CISTI). 2011 6th Iberian Conference on , pp.1-4, 15-18 June 2011.
[6] Katarzyna Keahey, Mauricio Tsugawa, Andrea Matsunaga, Jose Fortes, Sky Computing. IEEE Internet Computing, pp. 43-51, September/ October, 2009.
[7] http://sla-at-soi.eu/
[8] Wolfgang Theilmann, SLA@SOI framework architecture – evaluated architecture. Deliverable D.A1a, 29/07/2011.
[9] http://www.irmosproject.eu/ [10] Andreas Menychtas, Final version of IRMOS overall architecture. Deliverable D3.1.4, 18/01/2011.
[10] http://www.jclouds.org/index.html
[11] http://karaf.apache.org/
[12] http://incubator.apache.org/deltacloud/index.html
[13] Michael Jakl, REST - Representational State Transfer. University of Technology of Vienna, 2005.
[14] http://fog.io/1.0.0/
[15] http://libcloud.apache.org/index.html
[16] http://4caast.morfeo-project.org/
[17] Eduardo Oliveros, Conceptual model and reference architecture. Deliverable D1.2.2, 09/09/2011.
[18] International Journal of Engineering and Advanced Technology (IJEAT)Volume 8 Issue Issue-6S3 Pages 2164-2166
[19] International Journal of Engineering and Advanced Technology (IJEAT)Volume 8 Issue Issue-S3 Pages 2071-74