

University of Piraeus

Department of Digital Systems

School of Information and Communication Technologies

Programme “Technology Education & Digital Systems”

MSc Dissertation

**Modeling and Measurement of Cloud Services
Performance**

Athanasia-Charalampia Evangelinou

Supervisor: Dimosthenis Kyriazis

November 2014

Piraeus

Πανεπιστήμιο Πειραιώς

This thesis is dedicated to my mother Panagiota and my godmother Maria
for their endless love, support and encouragement

Πανεπιστήμιο Πειραιώς

Acknowledgements

First and foremost I want to thank my supervisor Dimosthenis Kyriazis. I appreciate all his contributions of time, advice, ideas and guidance to complete this work.

I would not have studied this MSc without the wise counsel of my professor Andriana Prentza who has been steady hands to steer me with patience through my undergraduate and postgraduate studies. Also, special thanks to my professor Marinos Themistocleous for his guidance and support during my years as a postgraduate student. I would like to express my special appreciation and thanks to my colleague George Kousiouris. His advice on my research has been invaluable.

I am indebted to all my friends who have supported me over the last few years and with whom I have enjoyed many useful and entertaining discussions: Nikos Chelmis, Leonidas Katelaris, Christina Santzaridou and Panagiotis Nikitopoulos.

Lastly, I would like to say a heartfelt thank you to my parents, my sister Anna and my godmother Maria for always believing in me and encouraging me to follow my dreams.

Abstract

Cloud services are emerging today as an innovative IT provisioning model, offering benefits over the traditional approach of provisioning infrastructure. However, the occurrence of multi-tenancy, virtualization and resource sharing in the cloud raise certain difficulties in providing performance estimation during application design or deployment time. In order to assess the performance of cloud services and compare cloud offerings from different cloud providers both the extension of an existing metamodel, namely CloudML@artist, for describing this information in a machine understandable format and cloud benchmarks are required. In this thesis context, both of these requirements have been implemented. Specifically, performance instances for different cloud providers are implemented based on CloudML@artist metamodel and to complete the instance creation, the incorporation of a number of performance metric values in the concrete instances for different cloud providers is provided. Performance measurement is achieved through benchmarking process and performance results are demonstrated from three large commercial cloud providers, Amazon EC2, Microsoft Azure and Flexiant, in order to support the provisioning decisions of the cloud users.

Contents

Introduction	9
1.1 Problem Definition	10
1.2 Thesis Focus	10
1.3 Thesis Structure	11
Background	12
2.1 Key Concept: Cloud Computing	12
2.1.1 Definitions	12
2.1.2 Essential Characteristics	14
2.1.3 Service models	19
2.1.4 Deployment Models	20
2.1.5 Taxonomies for available Cloud Services	21
2.1.6 Popular Cloud Providers and Services	22
2.2 Key Concept 2: Performance	25
2.2.1 Performance metrics needed in Cloud environments	26
2.2.2 Metrics for variation	28
2.2.3 Stereotypes and extraction of performance characteristics	29
2.2.4 Performance aspects of Cloud computing	30
Cloud Benchmarking	32
3.1 Definition and requirements of benchmarking	32
3.2 Standards bodies defining benchmarks	34
3.3 Existing benchmarks	36
3.3.1 Frameworks for comparing cloud providers	36
3.3.2 Application Benchmarks	39
3.3.3 MATLAB Benchmark suite	40
3.3.4 Berkeley Dwarfs suite	41
3.3.5 FileBench	41
3.3.6 Cloudsuite	42
3.3.7 DaCapo	43
4.1 Key Concept: UML Profiles	48
4.1.1 Models	48

4.1.1.1 Making up a model.....	50
4.1.2 Meta-Object Facility.....	51
4.1.3 UML 2.0 and Profiles package	51
4.2 Key Concept 2: Model-Driven Approach to Cloud	53
4.2.1 Model-Driven Engineering in a Nutshell	53
4.2.2 Model-Driven Architecture	54
4.2.3 Model-Driven Modernization.....	55
4.2.4 MDA and UML Profiles.....	56
4.2.5 Available Standards/Profiles/MetaModels of the chosen technology	56
4.2.5.1 REMICS PIM4Cloud and CloudML.....	56
4.2.5.2 CloudML: Cloud Modeling Language	59
4.2.5.3 SysML	61
4.2.5.3 fUML.....	64
4.2.5.4 Cloud4SOA semantic model	68
4.2.5.5 Blueprint Template	70
4.2.5.6 CloudML@artist.....	72
CloudML@artist Implementation	73
5.1 Why CloudML@artist	73
5.2 UML Profiles Description and meta-model Structure	74
5.3 Overall Process for Instance Creation.....	82
5.3.1 Installation of CCloudML@artist.....	82
5.3.2 Amazon EC2 Instance creation	82
5.3.3 Benchmarking process for performance results collection.....	84
Benchmarking Case Study On Three Selected Cloud Providers: Amazon EC2,Microsoft Azure and Flexiant.....	88
6.1 Benchmarking process.....	88
6.2 Benchmarking Results.....	90
Conclusions and Future Work.....	93

Πανεπιστήμιο Πειραιώς

CHAPTER 1

Introduction

Cloud computing is the delivery of on-demand IT resources and services over the Internet by paying a monetary value only for the duration of the usage of resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing provides three main types of cloud service models that have emerged as an innovative IT provisioning model in the recent years and will be extensively analyzed in chapter 2: software-as-a-service (SaaS), platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS). In this sense cloud computing has the potential to change radically the mode of computing resource and application deployment making room for new business models.

Providers like Amazon, Google and Microsoft have been established as platform and infrastructure providers in the cloud computing market. Beside them there emerge more and more providers, who build their own applications or consulting services upon infrastructure services offered by other market players [2]. However, after their usage severe considerations have emerged with regard to their varying performance due to multi-tenancy and resource sharing issues.

These issues make it very difficult to provide any kind of performance estimation during application design or deployment time. The issue of provider performance should be taken very seriously especially during the migration process of an application to the Cloud in order to save money and to guarantee a stability in the migrated application.

In order to implement a successful migration of a legacy system on Cloud environments, one should take into account the specificities and characteristics of the target platforms. For doing so, a suitable metamodel framework should be used that take under consideration the nature and features of the latter that could be extended in order to measure them and describe them accordingly to the application modelling layer.

1.1 Problem Definition

The issues of Cloud environments instability with regards to performance issues of the allocated resources have begun to arise after Cloud Provider's promises for infinitive resources and on-demand scalability [3]. Different providers have their own metrics and strategies for guaranteeing Cloud QoS regarding the performance. In this case for identifying performance aspects of Cloud environments the need of a more abstracted and machine understandable way is required. In order to achieve this goal the implementation and the extension of an existing metamodel framework (which is an approach enabling rich abstracted description of cloud services), namely CloudML@artist for describing Cloud Providers is needed. This way simplifies the configuration of a well-defined SLA (Service Level Agreement) [4] and customers are able to assess and select cloud services according to their performance requirements.

Moreover the fact that cloud providers are separate entities and no information is available on their internal structure and operation, makes it necessary to macroscopically examine a provider's behaviour with regard to the offered resources and on a series of metrics. This process should be performed through benchmarking, by using the suitable tools and tests. The results from the benchmarking process for each cloud provider are incorporated in the CloudML@artist framework in order to provide a way to describe, measure and select the fittest Cloud services based on their features and characteristics and capabilities.

1.2 Thesis Focus

In existing research outcomes there is a significant gap regarding abstracted descriptions in current meta-models that are related to cloud infrastructures and are used in order to describe the features and capabilities of Cloud providers and services. The functional ones have been significantly covered through many efforts [5], however one of the main interesting features refers to the modelling and description of the performance of the service offerings. CloudML@artist is the only framework that includes the main functionalities that are offered by the cloud platforms so that they can be considered during migration to the service oriented final version of the application and is enriched with performance requirements of most prominent application types in order to ensure a successful migration.

The main purpose of this thesis is to extend this current meta-model description by incorporating a number of metric values in the specified performance metrics to the concrete instances of different cloud providers such as Amazon EC2, Windows Azure and Google App Engine. In order to collect the performance data, a set of third party benchmarking tools have been used and specific scripts have been developed. The identified metrics are added to the cloud provider models providing the ability to characterize a Cloud service's ability from performance point of view, compare and finally evaluate them. Regarding the benchmarking process of cloud providers, one of the key aspects that should be taken under consideration is that due to the dynamicity in resource management, benchmarking must be iterated over time, so that we can ensure as much as possible that different hardware, different management decisions (like e.g. update/reconfiguration/improvement of the infrastructure) are demonstrated in the refreshed metric values, but also observe key characteristics such as performance variation.

1.3 Thesis Structure

The remainder of the current thesis is as follows: Section 2 presents the key concepts, including cloud computing general information (e.g. characteristics, service and deployment models, etc.) and performance related material (e.g. metrics, characteristics, etc.), while Section 3 introduces benchmarking concepts that are core in this thesis. Profiles and models are discussed in Section 4 capturing UML2.0, model-driven, MDA and CloudML amongst others. Section 5 describes in detail the CloudML@artist metamodel and the reasons for which it was selected in order to be extended. Moreover, this section presents both the overall process for instance creation and the benchmarking process for performance results collection. An example for Amazon EC2 is provided in order to the procedure to be comprehended. Section 5 presents a benchmarking case study on three selected cloud providers: Amazon EC2, Microsoft Azure and Flexiant and results are demonstrated. Finally section 7 includes the conclusions and the future work.

Chapter 2

Background

2.1 Key Concept: Cloud Computing

Cloud computing [6] has emerged as a viable means for delivering IT services and assumes that every software application or system component becomes a service or part of a service. Over the last years IT professionals, business managers and researchers defined cloud computing differently according to their understanding of its offering. With cloud computing the resources such as processing power, storage space, bandwidth, memory and software are provided as general utilities that can be leased and released by users through the internet in an on-demand way. By this way the resources are shared and so are the costs. Cloud service providers offer a variety of service models and pricing schemes to customers in order to compare the cloud computing services and select an appropriate solution.

Cloud computing represents a convergence of two major trends in information technology [7]. The first one is IT efficiency which is related to the power of modern computers is utilized more efficiently through highly scalable hardware and software resources. The other trend is business agility, whereby IT can be used as a competitive tool through rapid deployment, parallel batch processing, use of compute-intensive business analytics and mobile interactive applications that respond in real time to user requirements.

2.1.1 Definitions

Cloud computing [8] is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility over a network.

Cloud computing [9] is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access

their personal files at any computer with internet access. This technology allows for much more inefficient computing by centralizing storage, memory, processing and bandwidth.

In [10] the formal definition of cloud computing is as follows: "It is an information technology service model where computing services (both hardware and software) are delivered on-demand to customers over a network in a self-service fashion, independent of device and location. The resources required to provide the requisite quality-of-service levels are shared, dynamically scalable, rapidly provisioned, virtualized and released with minimal service provider interaction. Users pay for the service as an operating expense without incurring any significant initial capital expenditure, with the cloud services employing a metering system that divides the computing resource in appropriate blocks.

According to [11] Clouds are considered as a pool of usable and accessible virtualized resources such as hardware, platforms and services. These resources are dynamically reconfigured to adjust to a variable load allowing also an optimal resource utilization. These resources are used by a pay-as-you-go model in which guarantees are offered by the Infrastructure Provider by means of customized SLA's.

The main reason of different perceptions of cloud computing is the fact that it is not a new technology but a new operations model that brings together a set of existing technologies to run business in a different way. Most of the technologies that cloud computing concept draw on such as virtualization and utility-based pricing are not new, however cloud computing leverages these existing technologies to meet the technological and economic requirements of today's demand for information technology [12].

NIST Definition

According to the official NIST(National Institute of Standards and Technology) definition [13], "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." The NIST definition lists five essential characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured service. It also lists three "service models" (software, platform and infrastructure), and four "deployment models" (private, community, public

and hybrid) that together categorize ways to deliver cloud services. The definition is intended to serve as a means for broad comparisons of cloud services and deployment strategies, and to provide a baseline for discussion from what is cloud computing to how to best use cloud computing.

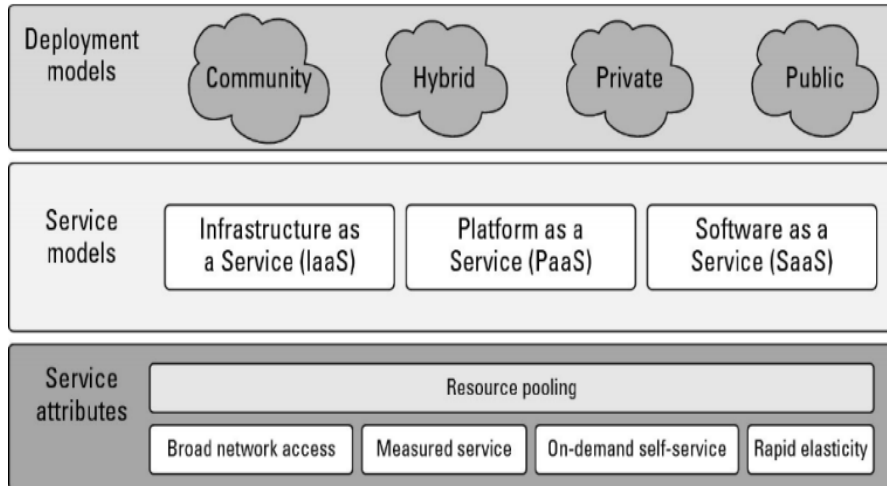


Figure 1: The NIST cloud computing definitions [112]

NIST definition of cloud computing 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.

2.1.2 Essential Characteristics

- **Characteristics by NIST**

On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

- **Additional characteristics[14]**

Scalability [15]: describes the system's ability to reach a certain scale. There are two approaches to scalability: scaling up that is achieved by providing more resources (more RAM, disk, virtual CPU etc.) and scaling-out (more machines or devices to the computing platform to handle the increased demand).

Vertical scaling (up) [16] can handle most sudden, temporary peaks in application demand on cloud infrastructures since they are not typically CPU intensive tasks. In scaling up the limitation is hardware related in a very specific: how much memory, disk and processor can be supported by a server. Horizontal scalability (scale out/in) replicates (or removes) instances of system elements (typically VMs) to balance the load.

Horizontal scalability [17] usually requires also the addition of another component that has the role of the Load Balancer (LB). Scaling out scalability is not automatic and must be architected into the system in other words it is an attribute of the architecture of the system. Horizontal scaling and load balancing are required for sustained increases in demand in order to restore and maintain peak performance.

Elasticity: According to [18] elasticity is the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. Systems have to autonomously execute predefined scalability actions to fulfill the contracted performance requirements with the minimum of resource demands [19]. The mechanisms and workflows that are used by the system to fulfill elasticity as well as the evaluation criteria and the decision-making process itself varies from one system to the other or from one application to the other, even in the same system.

Virtualization [20] is not a new concept and is the main enabling technology for cloud component, which uses a physical resource such as a server and divides it into virtual resources called virtual machines. There are 6 major types of virtualization: hardware, software, memory, storage, data and network virtualization. Virtualization is the key to cloud computing, since it is the enabling technology allowing the creation of an intelligent abstraction layer which hides the complexity of underlying hardware or software.

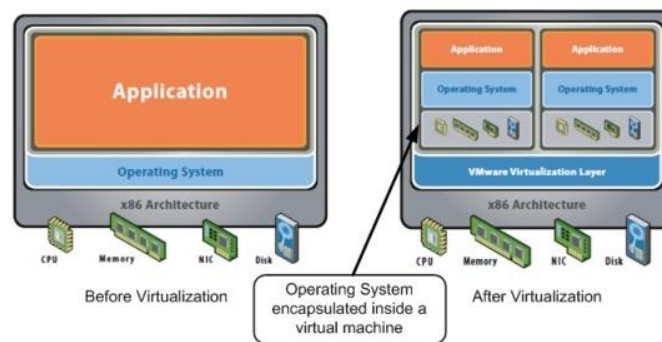


Figure 2: Virtualization into layered architectures [113]

Server virtualization is the moving of existing physical servers into a virtual environment, which is then hosted on a physical server. This type is where most of the attention is focused right now in the world of virtualization and is where most companies begin an implementation of this technology. Many modern servers are able to host more than one server simultaneously, which allows you to reduce the number of servers you have in your company, thus reducing your IT and administrative expenditures. Some servers can also be virtualized and stored offsite by other hosting companies.

Software virtualization (hypervisor) abstracts the software installation procedure and creates virtual software installations [21]. It emulates computer and allows different operating systems to run on a single physical computer host. Each of the guest operating system seems to have the host's processor, memory, and the other resources all to itself. The hypervisor, however, is actually controlling the host processor and resources and allocates what is needed to each OS, making sure that the virtual machines cannot disrupt each other. There are two types of hypervisors hosted hypervisors and bare metal or native hypervisors. Hosted hypervisors are run as a software using an operating system while bare metal hypervisor run on the host's hardware in order to control it and also to manage the guest operating systems.

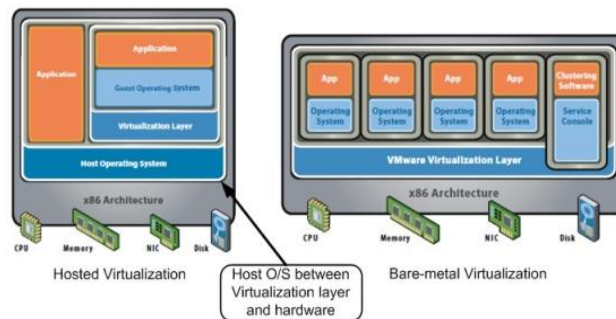


Figure 3: Bare-metal and hosted virtualization types [113]

Hardware virtualization [22] is achieved by abstracting the physical hardware layer by using a hypervisor. The hypervisor handles sharing the physical resources of the hardware between the guest operating systems running on the host. Physical resources become abstracted versions in standard formats, so regardless of the hardware platforms, the hardware is presented as the same model. The virtualized operating system is able to hook into these resources as though they are physical entities.

Storage virtualization [23] is a major component in storage for servers, in the form of controllers and functional RAID levels **Error! Reference source not found.** Operating systems and applications with raw device access prefer to write directly to the disks themselves. The controllers configure the local storage in RAID groups and present the storage to the operating system as a volume (or multiple volumes, depending on the configuration). The operating system issues storage commands to the volumes, thinking that it is writing directly to the disk. However, the storage has been abstracted and the controller is determining how to write the data or retrieve the requested data for the operating system.

Memory virtualization [25] is seen as virtual memory, or swap, on servers and workstations. Theoretically, swap is used when the amount of physical memory is full. The host sees the local swap as additional addressable memory locations and does not delineate between RAM and swap. In the same way as swap, memory virtualization allows networked, and therefore distributed, servers to share a pool of memory to overcome physical memory limitations.

Data virtualization [26] is any approach to data management that allows an application to retrieve and manipulate data without requiring technical details about the data, such as how it is formatted or where it is physically located. Managing data location and availability can be difficult when trying to pull from many sources to analyze the data. Data virtualization deals with the ability to abstract the actual location, access method and data types, and allow the end user to focus on the data itself.

For **network virtualization** [27] this remains true, although not so clearly as server virtualization. Networking devices utilize both paravirtualization and hypervisor techniques.

The first is loosely based on the idea of paravirtualization, where the underlying software is creating a separate forwarding table for each virtual network, such as is done by MPLS within each VRF. In MPLS, the OS creates a single routing and forwarding database for each VRF, but marks each entry in the database with the tag for ownership. BGP is used to update the database, and shares the routes AND the tags to distribute the data throughout the network.

In the second type of hypervisor, the network device OS instantiates multiple instances of the OS. Perhaps the most common example of this might be Cisco ASA firewalls, with the use of Virtual Contexts. Each context appears as a totally separate ASA instance and shares access to the physical interfaces. No communication between contexts is possible within the ASA OS, and all traffic must pass on physical interfaces.

Reliability[28]: is related to the reassurance that a system will perform its intended function for the required duration within a given environment, including the ability to test and support the system through its total lifecycle. For software, it defines reliability as “the probability of failure-free software operation for a specified period of time in a specified environment Users will

expect the cloud to be a reliable resource, especially if a cloud provider takes over the task of running “mission-critical” applications and will expect clear delineation of liability if serious problems occur.

Multi-tenancy[29]: is an architecture in which multiple users who do not share or see each other’s data can share the same applications while running on the same operating system, using the same hardware and the same data storage mechanism. In cloud computing, the meaning of multi-tenancy architecture has broadened because of new service models that take advantage of virtualization and remote access.

2.1.3 Service models

The services of cloud computing are broadly divided into three categories. Each category serves a different purpose and provides different offers for business and individuals.

Software as a Service (SaaS) [30] is a software model in which applications are hosted by cloud provider and are accessible from various client devices through web browser or an interface. This is a “pay-as-you-go” model and its advantage is that no need of specific hardware to run software, pay per use instant scalability, security and reliability. Also the user is not responsible for the the management and the controlling of the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities

Platform as a Service (PaaS) [31]. The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment. The advantages of PaaS [32] are no need to buy special hardware and software to develop and deploy enterprise applications, pay per use, instant scalability, security, reliability; the popular services are storage, database and scalability. Some examples are Google Apps and Microsoft Windows Azure.

Infrastructure as a Service (IaaS) [33]. The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The user does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and a limited control of select networking components. The advantages of IaaS are pay per use, instant scalability, security, reliability and APIs.

Apart from the aforementioned categories in [34], an approach of a novel cloud layer called Hardware-as-a-Service (HaaS) is described. HaaS focuses the transparent integration of remote hardware that is distributed over multiple geographical locations into an operating system. The local system will appear as if all hardware devices are connected locally. This model is advantageous to the enterprise users, since they do not need to invest in building and managing data centers. Potentially, everything from generic word processing software to customized computer programs designed for a specific company could work on a cloud computing system.

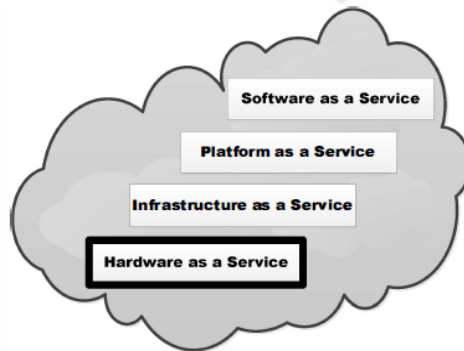


Figure 4: Traditional cloud stack extended by the novel HaaS cloud layer [34]

2.1.4 Deployment Models [13]

Private cloud. The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud. The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the

organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud. The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud. The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

2.1.5 Taxonomies for available Cloud Services

Regarding the three main categories of cloud services many research efforts have gone into clearly defining a taxonomy of cloud computing. One of the most important advantage of taxonomies is that they provide a common terminology to facilitate understanding and communication. Intel has created the Cloud Computing Services Taxonomy [35]. This classification consists of some primary categories of cloud computing services. Each of the categories is further divided into different subcategories. The authors in both [36][37] are suggesting an ontology describing the knowledge domain of cloud computing services. Another taxonomy for cloud computing is created in [38] this time from the viewpoint of the enterprise and the consumers instead of the vendors. The taxonomy created by [39] is built up from the different common characteristics one can find within the services. The overview provided in [40] also gives a good indication of the different services available.

The taxonomy as provided by Intel Figure 5 can be used as a guidance tool since it extensively covers the breadth as well as the depth of existing cloud services.

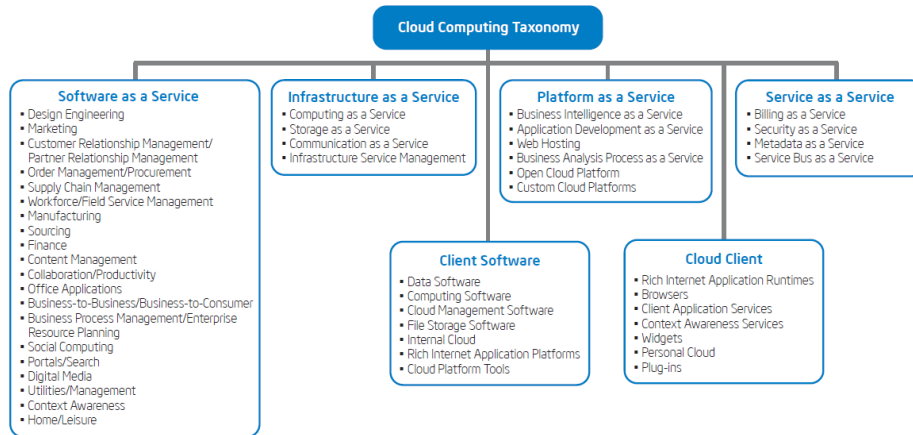


Figure 5: Cloud Computing Taxonomy [114]

The focus in this section is on the four main categories of services. Each of the main vendors/providers in their respective category, are discussed by means of the subcategories in the taxonomy. For the services discussed, also the available interfaces (i.e. (REST) API, components...) are given which provides us with an indication of how they can be used by other parts of an application.

2.1.6 Popular Cloud Providers and Services

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) [41] provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Amazon EC2, the basic service of the Amazon's cloud computing platform, provides a virtual computing environment for configuring, loading, monitoring and managing virtual machine instances originating from pre-configured, templated Amazon Machine Images (AMIs), or from images created and configured by user. Amazon EC2 works in conjunction with other Amazon services that are to be described (auto-scaling, load-balancing, storage, database, queuing) offering high scalability, failure resilience and security.

The following Amazon services enable scaling and load balancing for EC2 instances. Both are very important for increasing the performance of applications hosted on the cloud platform as well as lowering the costs.

- **Auto-Scaling** is the web service that applies a scaling action for scaling up or down on the selected EC2 instances, when the conditions have been met. Every one of these three variables (scaling policy, instances to be scaled and alarming conditions) are defined by the user. The conditions

on which the scaling takes place can be set on every metric available from the CloudWatch Amazon service (monitoring tool) such as disk utilization and network activity. In case there is a demand for condition stabilization before the triggering of the scaling action there is a variable that defines a cooldown time (waiting time) while the conditions continue to be scale-demanding before the alarm activation.

- **Elastic Load Balancing** is a service responsible for automatically distributing incoming traffic across multiple EC2 instances. Every load balancer has the ability to scale its request handling capacity according to the traffic growth. Except for handling incoming requests, load balancers contribute to the fault tolerance increment of the application by detecting non-healthy instances. When such an instance is found, the service routes no more traffic to this target. Interesting features of the Load balancing service are:
 - The ability to address the instances behind the load balancer locally using as external point only the public IP of the load balancer.
 - The combination of Load balancing and Auto-scaling for management purposes.
 - Traffic redirection to another destination if Load balancer or application instances seem to be unavailable (Amazon Route 53 Domain Name System web service)
 - AWS elastic load balancing provides the choice of sticky load balancing.

Microsoft Azure

Microsoft Azure [42] is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters. It provides both PaaS and IaaS services and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

The Compute Service of Microsoft Azure is provided in both infrastructure and platform levels.

Concerning Infrastructure as a Service, Windows Azure provides the ability to host application components on virtual machine instances. Some important information about these virtual machines is:

- **Creation:** There are two ways of creating virtual machines. Either perform direct creation by selecting an image

provided in the Image Gallery of the Windows Azure Management Portal or upload an image created by the user. The Image Gallery contains images with a variety of operating systems (several Linux distributions, Windows and Mac).

- **Scaling:** Virtual machines can be automatically scaled by increasing or decreasing the number of virtual-machine instances that are used by the application. The scaling can be configured based on Average CPU usage and the number of messages in a queue.
- **Load balancing** VM instances can be implemented in two ways. The first one is by creating a load balanced endpoint (TCP or UDP endpoint) used by all VM instances contained in a cloud service. The endpoint can provide round-robin load balancing. The second one is by using **Traffic Manager** to perform load balancing following one of the three available methods per policy: performance (traffic routed to the geographically closest hosted service), failover (traffic is routed to the first in order working hosted service), round-robin. Traffic Manager works by applying an intelligent policy engine to the Domain Name Service (DNS) queries on domain name(s).

Concerning Platform as a Service, Windows Azure provides the **Cloud Service**. A cloud service consists of the code and the configuration of an application hosted in Azure's platform. The infrastructure, the instances and the operating systems are maintained by Azure, while service upgrades are enabled without any interruption in service. Two types of role concepts are defined:

- The web role is a dedicated web-server used for hosting front-end applications.
- The worker role is used for serving requests in the background, independently of user interaction or input, and thus allows asynchronous execution as well as long-running or perpetual tasks.

Cloud Services run on role instances. One or more role instances can be used for a specific role. For scaling purposes, the number of role instances can be increased or decreased according to the scaling demands.

Google App Engine

GAE [43] is a web application hosting service, allowing for development and deployment of web-based applications within a pre-defined runtime environment. Unlike other cloud-based hosting offerings such as Amazon Web Services that operate on an IaaS level, the GAE already provides an application infrastructure on the PaaS level. This means that the GAE abstracts from the underlying hardware and operating system layers by providing the hosted application with a set of application-oriented services. While this approach is very convenient for developers of such applications, the rationale behind the GAE is its focus on scalability and usage-based infrastructure.

Flexiant Cloud Provider

Flexiant[44] is a UK-based software company that provides software to cloud services providers. Flexiant's cloud management software suite gives cloud service providers the ability to rapidly design and launch commercial cloud services by enabling business agility and flexibility to scale, deploy and configure cloud services, simply and cost-effectively. **Flexiant Cloud Orchestrator** is a fully automated software suite that enabled managed service providers, hosting providers, data centre operators and enterprises to offer cloud-computing services to their customers.

2.2 Key Concept 2: Performance

Performance is an attribute of cloud environments that has started to get significant attention in the recent years. Performance [16] is generally tied to an application's capabilities within the cloud infrastructure itself. Limited bandwidth, disk space, memory, CPU cycles, and network connections can all cause poor performance. In some cases, poor application performance is a combination of lack of resources, while in some others is an application architecture that does not properly distribute its processes across available cloud resources. Stakeholders in cloud such as infrastructure providers, software service providers, and end users have different performance concerns. Infrastructure providers, give emphasis to the utilization of the resources by meaning that they are interested in releasing timely resources so that the system can re-allocate to other applications and customers. From the service providers' perspective, it needs to balance between system performance and cost of resource reservation. If resources are reserved more than needed,

they have to pay for wasteful resources. If resources are reserved less than needed, they cannot guarantee service availability and response time.

2.2.1 Performance metrics needed in Cloud environments

Generic performance metrics

Currently, there are many companies moving their entire applications or part of them to the Cloud; thus, performance measurements (e.g. the percentage of CPU allocated to VM, disk IO, cache sharing, scheduling granularity, memory access patterns, etc.) should be taken into consideration, as different types of applications have different interference effects in distributed and virtualized environments. Therefore, organizations should choose a cloud service that will provide good throughput and low latency to their customers in order to avoid load imbalance and scalability and data management concerns. Given that performance issues are considered unpredictable in Cloud, this is considered one of the major obstacles in implementing reliable cloud services.

In [45] some vital performance metrics are identified such as:

Benchmark finishing time: this metric measures how long the instance takes to complete the benchmark tasks that stress each of the main compute resources (CPU, memory, and disk I/O).

Scaling latency: It's the time which needs a provider to allocate a new instance when customer requests it. Scaling latency can affect the performance and cost of running an application.

Persistent Storage: There are three common types of storage services which cloud providers offer for application state and data: table, blob and queue. The cloud storage has two advantages: scalability and availability. We use three metrics to compare the performance and cost of storage services: operation time, time to consistency, and cost per operation.

Operation response time: This metric measures how long it takes for a storage operation to finish.

Intra-cloud Network: The intra-cloud network connects a customer's instances among themselves and with the shared services offered by a cloud.

To compare the performance of intra-cloud networks common metrics are path capacity and latency.

Wide-area Network: includes the collection of network paths between a cloud's data centers and external hosts on the Internet. For the hosting of customer applications there are multiple locations so that requests from an end user can be served by an instance close to that user to reduce latency.

According to [46] some more interesting performance metrics are included in the table below (Table 1):

Sl. No.	Performance metric	Description
Scalability based metrics		
1	CPU capacity	CPU's speed in flops
2	Memory size	In general, cache memory size for a VM
3	Scale up	Maximum number of VMs allocated for an user
4	Scale down	Minimum number of VMs allocated for an user
5	Boot time	Booting time for a VM to get ready for usage
6	Storage capacity	Storage size of data
7	Scale uptime	Time taken for increasing a specific number of VMs
8	Scale downtime	Time taken for decreasing a specific number of VMs
9	Autoscale	Boolean value for autoscaling feature
10	Response time	Time required to complete and receive a process
Architecture specific metrics		
11	Pipeline stalls	Processor specific pipelines stalls e.g. IA64

12	Cache misses	L2 or L3 cache misses
13	Frequent voltage switches	Voltage variations caused due to applications in processors such as Nehalem

Table 1: Performance metrics [46]

Other interesting metrics that have been identified in[47] are shown below:

1. Memory speed more important for data-intensive applications such DBMSs or MapReduce
2. Disk I/O (sequential and random) many cloud applications require instances to store intermediate results on local disks if input data may not be processed in main memory.
3. Network bandwidth and GPU load between instances because application exchange through the network large amounts of data.
4. Data processed per second and data processed per Joule in order to evaluate the entire cloud system for large data applications.

2.2.2 Metrics for variation

According to the results of a worthwhile research [48] both small (corresponding to 1.76B of main memory 1 ECU) and large (corresponding to 7.4 GB of main memory, 4ECU) instances suffer from a large variance in performance and this is a consequence of the different system types used by virtual nodes. Also runtime measurements on the cloud suffer from high variance and are repeatable to a limited extent. Given that Cloud users need stability in the performance of their resources (e.g. scientists to reproduce the results or enterprises to allocate the suitable amount of resources for their applications and guarantee QoS to their customers), metrics for the stability of each observed metric that should be taken into consideration may be:

- Repeated measurement process over time to observe variations of offerings
- Standard deviation of the measured metric on a Cloud provider offering
- Probabilistic distribution information (e.g. confidence intervals, type of distribution)

2.2.3 Stereotypes and extraction of performance characteristics

The main goal of the performance stereotypes [51] is to extract a number of performance characteristics of the provider that are necessary for meeting QoS requirements in the migrated cloud applications. In order to achieve this, a concrete set of these characteristics must be defined so that they can be measured. The source of these characteristics is threefold:

- a) the generic application types that exist in modern software creation (like application servers, DB servers, mathematical computations etc.). These types indicate different patterns of usage of the underlying physical resources (mainly CPU, storage and networking) by each application type (e.g. random read operations in contrast to sequential write operations, floating point calculations vs matrix multiplication, regular or irregular memory accesses etc.). That information regarding these potential classes of applications could come from existing benchmark tests (targeting at specific types) or categorizations (TPC, Berkeley Dwarfs etc.). The basic output of this process will be the concrete categories of computational profiles that match common application types
- b) The specific features of the cloud environment that could affect application performance and operation. These may be derived from the nature of cloud computing and scalable resources. Examples of these features are scalability capabilities, elasticity delays (e.g. how fast a VM can be started and included in the service), security capabilities (e.g. resilience in different types of DoS attacks). These features also need consideration given that the providers do not share information on their management and configuration aspects, so the only way of determining their abilities is through a macroscopic observation (e.g. launching a DoS attack and measuring metrics such as server response degradation etc.)
- c) The specific cloud services that are investigated in Chapter 2. These services may have specific metrics that need to be observed and can be based on a per case examination. For example, the ability to launch specialized services like MapReduce clusters may come with its own performance test (e.g. Terasort). The same applies for specific purpose services like monitoring, billing etc. The performance of these services may directly affect application performance (e.g. if the

application elasticity is based on billing or monitoring information regarding the resources and the delay of such services is restricting).

2.2.4 Performance aspects of Cloud computing

The main performance aspects of Cloud computing can be summarized as follows:

- a) Heterogeneous and unknown hardware resources: the computing resources offered by the cloud providers are unknown to the external users. Available information may be limited to number of cores for example, memory sizes or disk quotes. However this level of information is far from sufficient in order to characterize the provider's hardware capabilities that may depend also on architecture, interconnection, RAM speeds etc. According to a study on Amazon platform conducted by Aalto University [49] the variation between the fast instances and slow instances can reach 40%. In some applications, the variation can even approach up to 60%. Identifying benchmarks that are not affected by hardware heterogeneity is a challenge and needs to be considered while deciding on the benchmarks.
- b) Different configurations: even in the existence of the same hardware however, the way this resource is configured plays a significant role in its performance. The same applies for software configurations (e.g. a DB instance over a virtual cluster) or variations in the software development. For example the key-value store Cassandra when tested with a specific configuration in a private cloud for a legacy application and when ported to public cloud with a Cassandra as a SaaS service might lead to performance variations based on the configurations of SaaS service.
- c) Multi-tenancy and obscure, black box management by providers: one of the main performance issues in cloud infrastructures is the fact that they deal with multiple different users that may start their virtual resources on the same physical host at any given time. However the effect of concurrently running VMs for example [48] significantly degrades the actual application performance. This is even more affected by the usage patterns of these resources by their virtual owners or their clients. Furthermore, consolidation decisions made by providers and that are unknown to the users may group virtual resources on the same physical node at any given time, without informing the owner.

- d) VM interference effects. In [50] an interesting research investigates the performance interference for a number of applications in experimental virtual environments that were selected for classifying their behavior using different metrics. Moreover a mechanism for the prediction of expected performance scores of the applications running different types of workloads was developed. The mechanism is able to predict scores with average error almost 5%. The environment which was developed was an experimental virtual resource allocation environment (VRA) and the examined applications in the two VMs were the compression, the compilation of source code, and rendering frames. The result from the research shows that combined performance varies substantially with different combinations of applications. Applications that rarely interfere with each other achieve performance to the standalone performance. However, some combinations interfere with each other in an adverse way. Moreover application scores are affected by different workloads and also by background applications. Finally from the findings about performance interference and workload characteristics was generated application clustering. Researchers ran a hierarchical clustering algorithm which used each application's performance score vector, which consists of normalized performance scores of an application against all the background applications. The application clusters are useful for predicting performance of a new application.
- e) Virtualization is a technology used in all cloud data centers to ensure high utilization of hardware resources and better manageability of VMs. According to study [51] despite the advantages provided by virtualization, they do not provide effective performance isolation. While the hypervisor (a.k.a. the virtual machine monitor) slices resources and allocates shares to different VMs, the behaviour of one VM can still affect the performance of another adversely due to the shared use of resources in the system. Furthermore, the isolation provided by virtualization limits the visibility of an application in a VM into the cause of performance anomalies that occur in a virtualized environment. Specifically, a user running the same virtual machine on the same hardware at different times will see wide disparity in performance based on the work performed by other VMs on that physical host.

Chapter 3

Cloud Benchmarking

The widely adoption of Cloud Computing technology triggered the need for Cloud Benchmarks in order to assess the performance of Cloud infrastructures and software and facilitate Cloud users' decisions for comparing Cloud offerings.

3.1 Definition and requirements of benchmarking

Traditionally benchmarks [52] are tools for providing a method of comparing the performance of various subsystems across different system architectures and answering which one is the best in a given domain. Most of these benchmarks require that the system under test is deployed in a managed environment using a fixed configuration. This means that the results coming out from benchmarking tests reflect the average performance of a static non changing system. Moreover each benchmark applies a representative scenario for the given domain taking into consideration the properties and constraints of the system to be benchmarked.

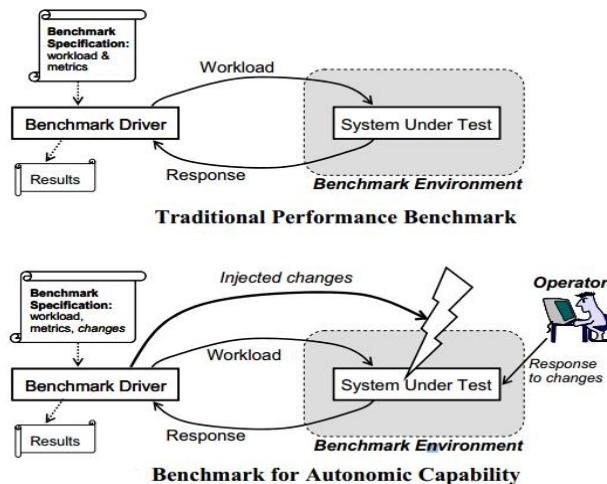


Figure 6: In a traditional performance benchmark a SUT is deployed in a stable benchmark environment and subjected to a synthetic workload designed to be representative of typical system use and it is needed to be extended in order to quantify the automatic characteristics of the SUT [98].

A critical aspect regarding benchmarking of one or more components of a multitier application is the System Under Test (SUT) [54] which is included in the benchmark definition. The SUT includes components whose performance is to be measured and exclude external systems that the application depends on but are not part of the performance evaluation. However, benchmarks measure the overall performance of the system because the isolated information about the component of interest demands the complete knowledge about all components involved.

According to [55] the most significant requirements result from a good survey on different benchmarking criteria are presented in table

Requirements	Description
General Requirements	this group contains generic requirements
(a) Strong Target Audience	the target audience must be of considerable size and interested to obtain the information
(b) Relevant	the benchmark results have to measure the performance of the typical operation within the problem domain
(c) Economical	the cost of running the benchmark should be affordable
(d) Simple	Understandable benchmarks create trust
Implementation Requirements	this group contains requirements regarding implementation and technical challenges
(a) Fair and Portable	all compared systems can participate equally
(b) Repeatable	the benchmark results can be reproduced by rerunning the benchmark under similar conditions with the same result
(c) Realistic and Comprehensive	the workload exercises all SUT features typically used in the major classes of target applications
(d) Configurable	A benchmark should

	provide a flexible performance analysis framework allowing users to configure and customize the workload
Workload Requirements	Contains requirements regarding the workload definition its interactions
(a) Representativeness	The benchmark should be based on a workload scenario that contains a representative set of interactions
(b) Scalable	Scalability should be supported in a manner that preserves the relation to the real-life business scenario modeled
(c) Metric	A meaningful and understandable metric is required to report about the SUT reactions to the load

Table 2: Benchmarking Requirements

By benchmarking in Cloud Computing we mean the testing process of services provided by different Cloud providers in which the SUT contains a Cloud service as component of interest. The main difference between the traditional and the Cloud Benchmarks is that for the latter we need different ways to measure performance and cost because in scalable systems the resources come and go. Moreover, a benchmark for the cloud should additionally test the cloud-specific features (scalability, pay-per-use and fault-tolerance) and provide appropriate metrics for them. The key challenge [56] of new benchmarks is to make the testing results comparable because different providers offer different services with different capabilities and guarantees of these services.

One of the key aspects is that due to this dynamicity in resource management, the benchmarking process must be iterated over time, so that we can ensure as much as possible that different hardware, different management decisions (like e.g. update/reconfiguration/improvement of the infrastructure) are demonstrated in the refreshed metric values, but also observe key characteristics such as performance variation, standard deviation etc.

3.2 Standards bodies defining benchmarks

Several consortia are defining standard domain-specific benchmarks,

standard price metrics, and standard ways of measuring and reporting results. The most prominent are:

Cloud Commons

The Cloud Service Measurement Index Consortium (CSMIC) [57] has identified metrics that are combined in the form of the Service Measurement Index (SMI), offering comparative evaluation of Cloud services. These measurement indices can be used by customers to compare different Cloud services.

SPEC (System Performance Evaluation Cooperative)

The Standard Performance Evaluation Corporation (SPEC)[58] is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC's Open Systems Group (OSG) has formed a new Cloud group to work in cooperation with other SPEC committees and subcommittees to define cloud benchmark methodologies, determine and recommend application workloads, identify cloud metrics for existing SPEC benchmarks, and develop new cloud benchmarks. OSGCloud working group was formed with the main goal to research and recommend workloads for cloud computing. The main goal of research benchmarks is to provide representative application scenarios, defined at a higher level of abstraction that can be used as a basis to evaluate early prototypes and research results as well as full-blown implementations of Cloud platforms.

The Working Group has identified three classes of interested parties to Cloud benchmark results: Hardware/Software-Vendors, Cloud-Providers and End-Consumers. These three parties form two distinct relationships which define two types of benchmarks: Black Box and White Box.

ETSI (European Telecommunications Standards Institute)

The European Telecommunications Standards Institute (ETSI) [59] produces globally-applicable standards for Information and Communications Technologies (ICT), including fixed, mobile, radio, converged, broadcast and internet technologies. ETSI is recognized by European Union as the European Standards Organisation.

The Perfect Club

A consortium of vendors and universities defining benchmarks for the scientific domain, with particular emphasis on parallel or exotic computer architectures.

TPC (Transaction Processing Performance Council)

A consortium of vendors defining benchmarks for transaction processing and database domains.

3.3 Existing benchmarks

In the following sections, a number of existing benchmarks are identified and highlighted, that may describe different applications, cloud providers in particular or service offerings.

3.3.1 Frameworks for comparing cloud providers

Various frameworks have been proposed for measuring cloud metrics and subsequently ranking the cloud services.

YCSB(Yahoo Cloud serving benchmark)[61] and AppScale**Error! Reference source not found.** are frameworks that are focussed on performance comparisons of distributed cloud serving datastores like Cassandra, MySQL etc.,

While AppScale uses DataStore API from Google AppEngine as a universal interface with various datastores and thus allowing various applications written for Google AppEngine to be tested with this framework without any modification. YCSB provides a DB Interface layer that translates simple requests to calls against the database (such as Thrift calls to the Cassandra or REST requests to PNUTS). YCSB focuses on transaction level access while AppScale on end-to-end application performance for DB accesses. YCSB measures the scalability and elastic speedup in addition to performance metrics like throughput, read latency etc., AppScale measures the end-to-end web application response time.

OpenBenchmarking.org[63], CloudHarmony[64], CloudSleuth[65] are performance measurement tools that archive the test results and make them available through the web. OpenBenchmarking.org is a comprehensive testing and benchmarking platform. It has an exhaustive list of test suites and test results on various hardware archived. Performance of hardware resources on various hardware can be consulted online from the archive. CloudHarmony provides a similar benchmarking solution but focused on cloud, and provides various performance metrics with focus on application, CPU, Disk I/O etc. for various cloud providers online.

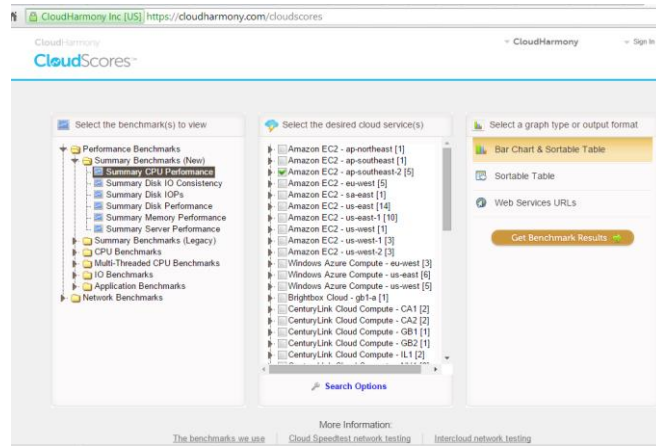


Figure 7: Selection of benchmark and desired cloud services in CloudSleuth [65]

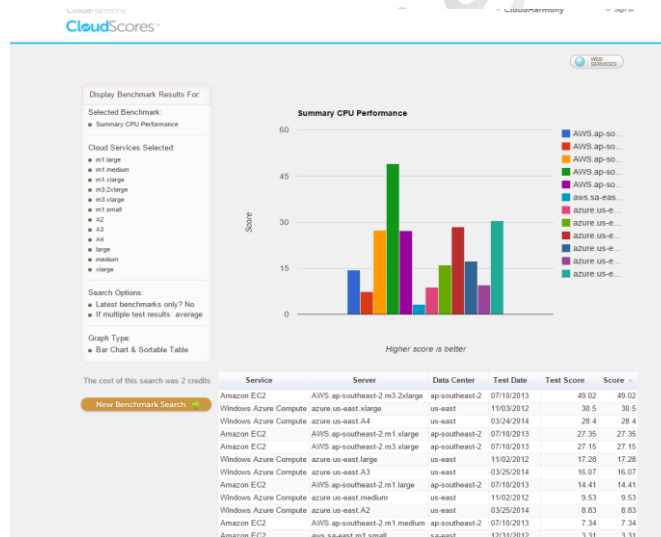


Figure 8: Performance results from CloudSleuth for the selected services [65]

The results of various runs are archived and are available for access through the web. CloudSleuth provides availability, response time of various cloud providers online by continuously monitoring a sample application running on top cloud computing providers.

CloudStoneError! Reference source not found. defines a benchmark to measure the performance of Web 2.0 applications on a cloud. It consists of 3 components – Olio(calendar application), Faban(Workload generator), Measuring and management tools. Olio is a social event calendar application, which can be deployed on the cloud system to be benchmarked. Faban is a workload generator that runs on the clients and simulates large number of

users simultaneously accessing Olio. Tools perform management tasks such as deploying Olio and measuring the performance of the cloud system. CloudStone could be useful to find the best system design for optimizing the architecture of a certain application.

CloudCmp [66] provides a methodology and as a goal has to estimate the performance and costs of a legacy application if it is deployed on a cloud provider. A potential cloud customer can use the results to compare different providers and decide whether it should migrate to the cloud and which cloud provider is best suited for its applications. CloudCmp identifies the common services for various cloud providers, and then for each service identifies a set of performance metrics relevant to application performance and cost, develop a benchmarking task for each metric and run the tasks on different providers and compare.

Skymark is a framework designed to analyze the performance of IaaS environments. The framework consists of 2 components – Grenchmark and C-Meter. Grenchmark is responsible for workload generation and submission while C-Meter consists of a job scheduler and submits the job to a cloud manager that manages various IaaS clouds in a pluggable architecture. Skymark [67] focuses on the low level performance parameters of Cloud services like CPU, Memory etc., Skymark does not consider the performance measurements of other cloud models like SaaS, PaaS and the metrics evaluated for IaaS performance comparison is not explicit.

SMICloud framework [69] provides a mechanism that measures the quality and prioritize cloud services. It defines a framework which consists of 3 elements – Service Catalogue, Monitoring, SMI cloud broker. Service Catalogue stores the services and the features as advertised by Cloud providers. Monitoring discovers Cloud services and monitors the performance of Cloud services and keeps track of how SLA requirements of previous customers are satisfied by Cloud provider. SMI Cloud Broker collects the customer application requirements and performs ranking of suitable services. It presents a QoS model for IaaS providers that can be extended for SaaS and PaaS in order to provide a comprehensive framework for cloud services measurement. SMICloud framework seems like an interesting starting point for ARTIST providing a layered approach for evaluating various cloud services and an initial list of metrics like cost, throughput etc., Various applications could specify the requirements and the framework ranks the cloud services based on the QoS attributes calculated from the current and historical performance data of various cloud services.

Cloud Rank D [70], is the first benchmark suite for evaluating cloud performance on the entire system's level, for large data applications. There are three ways of using the benchmark suite. The first one is that a user can quantitatively measure metrics of different cloud system and especially measure how much a system outperforms another one. The second one is that Cloud Rank-D can guide the optimization of a system under test and the third one is that we can rank different systems according to metrics derived from Cloud Rank-D. In addition, the Cloud Rank-D suite includes a set of 13 representative data analysis tools; thus, the users, according to their business requirements, can choose one of the four basic categories of benchmarks in Cloud Rank D: transformation, aggregation, summary, expansion and one derived category: hybrid. In order to create a benchmark application, due to the fact that the best programs to use for benchmarking are real or simply applications, the top-down method was used. The benchmark suite includes basic operations for data analysis, classification, clustering, recommendation, sequence learning, association rule mining and data warehouse operations as well as a real application which is called ProfSearch. Moreover the benchmark suite includes most popular data mining algorithms as naive Bayes support vector machine and k-means.

3.3.2 Application Benchmarks

PARSEC (Princeton Application Repository for Shared-Memory Computers) is a new benchmark suite for evaluating multi-core and multiprocessor systems of Chip-Multiprocessors (CMPs), that was released at the beginning of 2008. All benchmarks are written in C/C++. The suite includes a number of RMS (mining and synthesis) applications, as well as systems applications but also several leading-edge applications from Princeton University, Stanford University, and the open-source domain. Some requirements for a benchmark suite which PARSEC addresses are emerging workloads, multithreaded applications and diversion of applications [71].

Rodinia[72] is a multi-platform benchmark suite for heterogeneous computing and it is based on Berkeley's dwarf taxonomy. The suite consists of four applications and five kernels which target multi-core CPU and GPU platforms and also issues related to parallel communication patterns and synchronization techniques.

Benchmarking of HPC applications

Cloud computing with scalable virtualization technique solves issues regarding the execution of HPC applications [73]. These kinds of applications often raise load imbalance, scalability, data management and security concerns. Some examples of existing HPC cloud applications include

- High Energy Physics Domain: eg. BaBar application, DZero
- Geographic/Seismic Domain: eg. most of these application are data sensitive
- Electronics Design Community: e.g. Static Timing Analysis
- Media and Gaming Domains
- Large – scale Engineering Simulation Studies

Benchmarks for HPC applications performance

- HPL (High Performance Linpack) benchmark of High Performance Computing Challenge.
- Nas Parallel benchmarks (NPB) [74]: are used to evaluate the performance of parallel computers and are derived from computational fluid dynamics (CFD) applications and consist of five kernels and three pseudo-applications. The benchmark suite has been extended in order to provide new benchmarks for unstructured adaptive mesh, parallel I/O, multi-zone applications, and computational grids.

3.3.3 MATLAB Benchmark suite

MATLAB benchmarks [75] consist of six benchmarks that are used for both determining the hardware computational capability (test score) and characterizing types of workloads (test number). Tests include floating-point with regular or irregular memory accesses, data structures, mixed integer and floating point operations, 2-D and 3-D graphics. A research, which is based on MATLAB benchmarks, is presented in [48]. The analysis is related with a number of crucial parameters which effect on the performance of VMs such as CPU allocation percentages, real-time scheduling decisions and co-placement of VMs when they run applications in the same physical node, and they share infrastructure. Moreover a black box method is described based on genetically optimized ANNs to model and to predict the performance of an application.

3.3.4 Berkeley Dwarfs suite

A dwarf [76] is an algorithmic method that captures a pattern of computation and communication. The first seven Dwarfs which are used for High Performance Computing were inspired by Phil Colella who identified seven numerical methods important for science and engineering. Instead of traditional benchmarks Dwarfs are used to design and design parallel programming models and architectures. Some examples of examined applications are dense matrices or vectors, linear algebra, data mining and clustering, sparse linear algebra (finite element analysis and partial differential equation), spectral methods (fluid dynamics, quantum mechanics and weather prediction), N-body methods (molecular modeling, molecular dynamics and cosmology), structured grids with high spatial locality (image processing such as SRAD and physics simulations such as Hotspot), irregular grids (belief propagation and computational fluid dynamics), MapReduce (distributed searching, sequence alignment and parallel Monte Carlo simulations) and many more. The main advantage of Dwarfs is that they cover a very large range of application categories and capture their computational patterns.

3.3.5 FileBench

Filebench [77] is a very flexible file system and storage benchmarking tool. Basically it is an open source C frameworks that uses loadable workload personalities to allow easy emulation of complex applications. We have tested the last version and it's is resulted quick to set up and easy to use. Filebench includes many features to facilitate file system benchmarking:

- Multiple workload types support via loadable personalities.
- Ships with a library of more than 40 pre-defined personalities, including the ones that describe mail, web, file, and database servers behaviour. Workload personalities define the workload to apply to the system; they include tunables for scaling workloads to specific systems.
- Easy to add new personalities using reach Workload Model Language (WML) [78]
- Multi-process and multi-thread workload support.
- Configurable directory hierarchies with depth, width, and file sizes set to given statistical distributions.
- Support of asynchronous I/O and process synchronization primitives.
- Integrated statistics for throughput, latency, and CPU cycle counts per system call.
- Tested on Linux, FreeBSD, and Solaris platforms.

In Figure 9 an example of output concerning Fileserver emulator benchmark:

```

root@user# go_filebench
Filebench Version 1.4.9
12324: 0.000: Allocated 170MB of shared memory
filebench> load fileserver
12462: 2.869: FileServer Version 2.2 personality successfully loaded
12462: 2.869: Usage: set $dir=<dir>
12462: 2.869:      set $meanfilesize=<size>      defaults to 131072
12462: 2.869:      set $nfiles=<value>      defaults to 10000
12462: 2.869:      set $nthreads=<value>      defaults to 50
12462: 2.869:      set $meanappendsize=<value> defaults to 16384
12462: 2.869:      set $iosize=<size>      defaults to 1048576
12462: 2.869:      set $meandirwidth=<size> defaults to 20
12462: 2.869: (sets mean dir width and dir depth is calculated as log (width, nfiles)
12462: 2.869:      run runtime (e.g. run 60)
filebench> set $dir=/mnt
filebench> run 60
12462: 4.909: Creating/pre-allocating files and filesets
12462: 4.918: Fileset bigfileset: 10000 files, avg dir width = 20, avg dir depth = 3.1, 1240.757MB
12462: 5.280: Removed any existing fileset bigfileset in 1 seconds
12462: 5.280: making tree for fileset /tmp/bigfileset
12462: 5.290: Creating fileset bigfileset...
12462: 6.080: Preallocated 7979 of 10000 of fileset bigfileset in 1 seconds
12462: 6.080: waiting for fileset pre-allocation to finish
12466: 6.080: Starting 1 filereader instances
12467: 6.081: Starting 50 filereaderthread threads
12462: 7.137: Running...
12462: 67.142: Run took 60 seconds...
12462: 67.145: Per-Operation Breakdown
statfile1      128311ops    2138ops/s    0.0mb/s      0.0ms/op     2320us/op-cpu [0ms - 0ms]
deletefile1   128316ops    2139ops/s    0.0mb/s      0.2ms/op     2535us/op-cpu [0ms - 458ms]
closefile3    128323ops    2139ops/s    0.0mb/s      0.0ms/op     2328us/op-cpu [0ms - 0ms]
readfile1     128327ops    2139ops/s    283.8mb/s    0.1ms/op     2460us/op-cpu [0ms - 267ms]
openfile2     128329ops    2139ops/s    0.0mb/s      0.0ms/op     2332us/op-cpu [0ms - 2ms]
closefile2    128332ops    2139ops/s    0.0mb/s      0.0ms/op     2332us/op-cpu [0ms - 0ms]
appendfilerandl 128337ops    2139ops/s    16.6mb/s     0.1ms/op     2377us/op-cpu [0ms - 559ms]
openfile1     128343ops    2139ops/s    0.0mb/s      0.0ms/op     2353us/op-cpu [0ms - 2ms]
closefile1    128349ops    2139ops/s    0.0mb/s      0.0ms/op     2317us/op-cpu [0ms - 1ms]
wrtfile1      128352ops    2139ops/s    265.2mb/s    0.1ms/op     2601us/op-cpu [0ms - 268ms]
createfile1   128358ops    2139ops/s    0.0mb/s      0.1ms/op     2396us/op-cpu [0ms - 267ms]
12462: 67.145: IO Summary: 1411677 ops, 23526 ops/s, (2139/4278 z/w), 565mb/s, 393us cpu/op, 0.2ms latency
12462: 67.145: Shutting down processes
root@user#

```

Figure 9: Filebench output example

3.3.6 Cloudsuite

CloudSuite [79] is a benchmark suite for emerging scale-out applications. The second release consists of eight applications that have been selected based on their popularity in today's datacenters. The benchmarks are based on real-world software stacks and represent real-world setups.

The **Data Analytics** benchmark relies on using the Hadoop MapReduce framework to perform machine learning analysis on large-scale datasets. Apache provides a machine learning library, Mahout that is designed to run with Hadoop and perform large-scale data analytics.

The **Media Streaming** benchmark consists of two main components: a client and a server. The client component emulates real world clients sending requests to stress a streaming server.

The **Data caching** benchmark uses the Memcached data caching server, simulating the behavior of a Twitter caching server using the twitter dataset. The metric of interest is throughput expressed as the number of requests served per second. The workload assumes strict quality of service guaranties.

The **data serving** benchmark relies on the Yahoo! Cloud Serving Benchmark (YCSB). YCSB is a framework to benchmark data store systems. This framework comes with the interfaces to populate and stress many popular data

serving systems. Here we provide the instructions and pointers to download and install YCSB and use it with the Cassandra data store.

We use the GraphLab machine learning and data mining software for the **graph analytics** benchmark. We implemented TunkRank on GraphLab, which provides the influence of a Twitter user based on the number of that user's followers. Although GraphLab can perform distributed graph processing, in this document, we provide instructions for a single-machine setup. Instructions for cluster deployment can be found at the GraphLab website.

Software testing is a resource-hungry and time-consuming task that can leverage cloud computing. There are many applications that can potentially benefit from the abundance of resources in clustered systems. This benchmark tests Cloud9, an automated software-testing platform that parallelizes symbolic execution and scales on clusters of commodity hardware.

The **search benchmark** uses the Nutch search engine to benchmark the indexing process. It consists of a client machine that simulates real world clients, a frontend server to accept the client requests and send them to the index processing nodes.

Web serving is a fundamental application in any Internet-based service. We use CloudStone in CloudSuite to benchmark Web 2.0 applications.

3.3.7 DaCapo

The DaCapo benchmark suite [80] is designed to facilitate performance analysis of Java Virtual Machines, compilers and memory management. This benchmark suite is intended as a tool for Java benchmarking by the programming language, memory management and computer architecture communities. It consists of a set of open source, real world applications with non-trivial memory loads. The DaCapo suite consists of the following benchmarks:

- AVRORA: simulates a number of programs running on a grid of AVR micro-controllers
- BATIK: produces a number of Scalable Vector Graphics (SVG) images based on the unit tests in Apache Batik
- ECLIPSE: executes jdt performance tests for the Eclipse IDE
- FOP: parses/formats XSL-FO file and generates a PDF file

- H2: executes a JDBC benchmark using a number of transactions against a banking model application
- JYTHON: interprets pybench Python benchmark
- LUIINDEX: uses Lucene to index a set of documents
- LUSEARCH: uses Lucene to search of keywords over a data corpus
- PMD: analyzes a set of Java classes for a range of source code problems
- SUNFLOW: renders a set of images using ray tracing
- TOMCAT: runs a set of queries against a Tomcat server retrieving and verifying the resulting webpages

Benchmarks	Application Type	Resource focus	Implementation	License type	Metrics
YCSB	Cloud OLTP(online transaction processing) applications	cloud serving systems(latency, scaling)	Java	Open source, extensible-easy definition of new workloads, easy to benchmark new systems.	Online read/write access to data, latency of requests when the database is under load, scalability-elasticity
PARSEC	Computer vision, physical modeling, future media, content based search, deduplication, finance-	Multicore	C/C++	Open source, extendable- build and run workloads (applications)for user	Cache (miss rates) during load and store,traffic from cache

	multimedia				
Rodinia	Dwarfs-scientific/engineering-data mining	Multicore, GPU, memory band-width	OpenMP, OpenCL&C UBA	Open Source-expand Rodinia in future to cover the remaining dwarfs	Parallel communication&data access patterns, data-sharing characteristics, power consumption
HPL	Basic operation is based on vector primitives	Mflop/s (millions of floating point operations per second)	C (installed MPI and BLAS or VSIPL)	Open Source(will be extended),Linux	machine's frequency, in cycles per second, the number of operations per cycle
NAS Parallel	Multi-zone applications, computational grids	CPU, GPU	MPI, OpenMP, Java	Open Source ,Linux, has been extended to include new benchmarks	unstructured adaptive mesh, parallel I/O, multi-zone applications, and computational grids
Cloud Rank D	Large data applications	CPU	Ha-doop(version 0.20.2), Hive(version 0.6.0) and Mahoot(version	NA	Performance on the whole system level by to complementary

			n 0.6)		metrics: data processed per second and per Joule (memory, disk and network I/O)
Berkeley Dwarfs	Numerical, 2D graphs, 3D animation applications	CPU speed, cache size or RAM size	Numerical methods	Open source	Performance of virtualized hardware, or PaaS provider measure performance of many IaaS providers
MATLAB	Multimedia, scientific applications	CPU, RAM accesses	Matlab scripting	Open source	CPU allocation percentages, real-time scheduling decisions
CloudSuite 2.0 Benchmark suite	Data Analytics, Data serving, Media streaming, Software testing, Web search, Web serving	Architecture	multiple languages	CloudSuite 2.0 license	multiple metrics depending on application

Cloud-Stone	Web 2.0 applications	Web 2.0 applications	Java, php, ruby	Apache 2.0	dollars per user per month
Filebench	File System-emulation of mail, web, file and database servers	File system and storage I/O, CPU,CPU/OP	uses extensive Workload Model Language (WML)	Open source	Integrated statistics for throughput, latency and CPU cycle counts per system call, I/O trace, NFS trace and application trace
DaCapo	Java-based client and server side applications	performance analysis of Java Virtual Machines, compilers and memory management	Java	Open source	Response time

Table 3: The most interesting benchmarks and their features

Chapter 4

UML Profiles - Model Driven Engineering and Existing Models for Cloud Descriptions

With the emergence of cloud computing, Software-as-a-Service (SaaS) is becoming mainstream. Hereby, software (or parts of it) is hosted in cloud environments [81] and consumable over the network by different clients. As cloud computing aims at improving the quality of delivered services with respect to rapid elasticity and high availability, as well as reducing the costs of software operation by a “pay-as-you-go” pricing model, there is an increasing need to move legacy software into the cloud of services. However, the systematic and efficient modernization of legacy software to exploit current cloud-based technologies remains a major challenge. Such a paradigm shift implies fundamental changes to how software is modernized, delivered and sold.

Moreover, the increasing complexity of information systems is challenging the way software architects and engineers work. The rapid advancements in the field of ICT in recent years and the software applications running on different platforms entail the development of software solutions in a manner that is independent of the technology change. After initially being concerned more about the structure and quality of programming code, software engineers are now focusing their attention on the modelling aspects of the system development process. In order to achieve both, a succeed modernized migration to the Cloud and the alleviation of the undesirable effects of technology change, the usage of UML Profiles [82] and the incorporation of Model Driven Architecture and Model Driven Modernization [83] in the development of cloud services is needed.

4.1 Key Concept: UML Profiles

4.1.1 Models

Models [84] provide abstractions of systems which help deal with larger and more complex applications in simpler ways regardless of how they are implemented and distributed and whichever the final execution platform or technology used.

A model is a description of (part of) a system written in a well-defined language. A well-defined language is a language with well-defined form (syntax) and meaning (semantics), which is suitable for automated interpretation by a computer [85].

The Object Management Group (OMG) [86] is an international, open membership, computer industry standards consortium which is focused on modeling and model-based standards and provides specifications as a standard. This consortium defines several modeling languages among which UML (Unified Modeling Language) [87] is the most-used specification and the way the world models not only application structure, behavior and architecture, but also business process and data structure.

UML is a language for specifying, visualizing, constructing and documenting models of software systems, including their structure and design, in a way that meets all of these requirements. It is a general purpose modeling language that can be used with all major object and component methods and can be applied to all application domains and implementation platforms (J2EE-Java 2 Enterprise Edition-.NET).

However, in some cases a language that is so general is not proper in order to model applications of some specific domain. For instance, when the syntax or semantics of particular systems of the UML elements cannot describe specific concepts of particular systems, or when is needed to restrict or customize some of the UML elements which are usually general.

One possible approach that is defined from OMG is a set of extension mechanisms such as stereotypes, tagged values and constraints.

- **Stereotypes** extend the vocabulary of the UML by creating new model elements derived from existing ones but that have specific properties suitable for a specific domain. Each stereotype defines a set of properties that are received by elements of that stereotype.
- **Tagged values** are the properties for specifying key-value pairs of model elements, where keywords are attributes. They are a convenient way of adding information to an element in addition to that directly supported by UML.
- **Constraints** are the properties for specifying semantics or conditions that must be maintained as true for model elements. The aforementioned customizations are sets of UML extensions grouped into UML Profiles. A Profile is a collection of such extensions that together describe some particular modeling problem and facilitate modeling constructs in that domain. However a disadvantage of this

approach is that it may not provide such an elegant and perfectly fitting notations as may required for those systems.

4.1.1.1 Making up a model

According to OMG definition in order to make up a model a four-layered architecture is provided which is consisted of the following different conceptual levels: the instances (M0), the model of the system (M1), the modelling language (M2), and the metamodel of that language (M3).

Layer M0: Instances. The M0 layer corresponds to the real world models the running system and its elements are the actual instances that exist in the system.

Layer M1: The model of the system. The elements of the M1 layer are models. There is a strong relationship between the M0 and M1 layers. The elements of the M1 layer are classifications of elements of the M0 layer. Likewise, each element at the M0 layer is always an instance of an element at the M1 layer.

Layer M2: The model of the model (the metamodel). The elements of layer M2 are the modelling languages. Layer M2 defines the concepts that are used to model an element of layer M1. Just as there was a close relationship between layers M0 and M1 so there is a close relationship between M1 and M2 layers. Every element at M1 is an instance of an M2 element, and every element at M2 categorizes M1 elements. The model that resides at the M2 layer is called a metamodel.

Layer M3: The model of M2 (the meta-metamodel). Finally, layer M3 defines the concepts that can be used to define modelling languages.

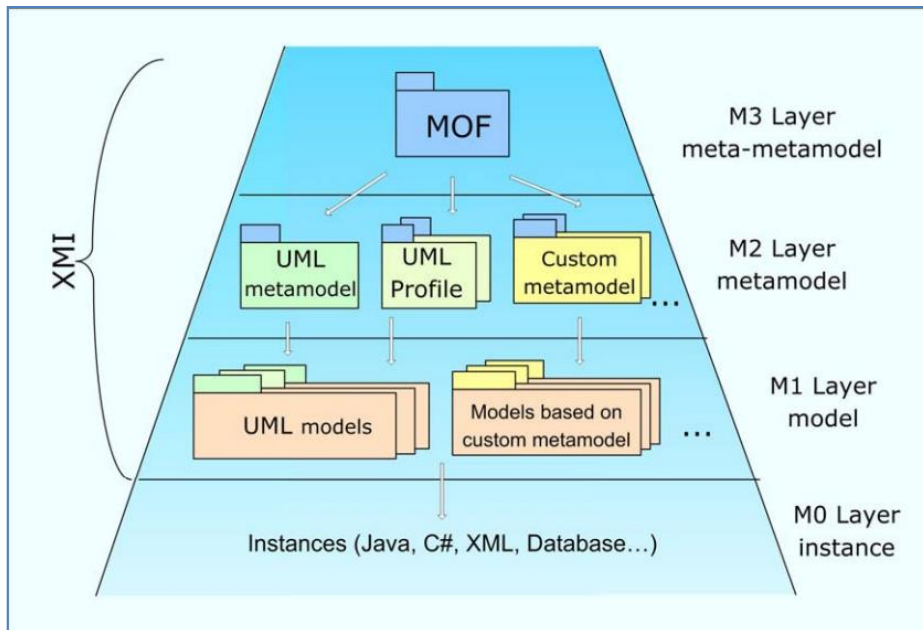


Figure 10: MDA four-layer MOF-based metadata architecture [115]

4.1.2 Meta-Object Facility

The modelling language defined for describing the M3 elements is called MOF (Meta-Object Facility) [88]. MOF is a Domain Specific Language (DSL) used to define modelling languages, such as UML, CWM, or even MOF itself. Such languages can be considered as instances of MOF. A well-defined language (such as UML) can be described by its metamodel. A model that represents a modeling language is called metamodel. What MOF provides is a language to describe metamodels. If we wanted to define a new object-based visual language other than UML we would use the MOF to describe its metamodel.

4.1.3 UML 2.0 and Profiles package

In order to define a new model language for the description of a system, UML can be easily customized by using a set of extension mechanisms that UML itself provides. UML 2.0 includes the Profile package that defines a set of UML artefacts. More precisely, the Profiles package included in UML 2.0 defines a set of UML artefacts that allows the specification of an MOF model to deal with the specific concepts and notation required in particular application domains (e.g., real-time, business process modelling, finance, etc.) or implementation technologies (such as .NET, J2EE, or CORBA). It should be

noted that UML Profiles allow the customization of any MOF defined (not just UML defined) metamodel. Similarly, a UML Profile can also specify another UML Profile.

UML Profiles

A UML Profile is defined as a UML package stereotyped profile that can extend either a metamodel or another Profile. As it was mentioned in UML section UML Profiles are defined in terms of three basic mechanisms: stereotypes, constraints, and tagged values.

The process for adding a new element in a UML model is the following:

- 1) First a stereotype is defined by a name and by the set of metamodel elements it can be attached to. Graphically, stereotypes are defined within boxes, stereotyped “stereotype”. Metamodel elements are indicated by classes stereotyped «metaclass». The notation for an extension is an arrow pointing from a stereotype to the extended class, where the arrowhead is shown as a solid triangle.
- 2) Constraints can be associated to stereotypes, imposing restrictions on the corresponding metamodel elements. In this way a designer can define the properties of a well-formed model. Constraints can be expressed in any language, including natural language or the OCL (Object Constraint Language). OCL is a language, now part of UML, adopted by the OMG for expressing constraints and properties of model elements.
- 3) Finally, a tagged value is an additional meta-attribute that is attached to a metaclass of the metamodel extended by a Profile. Tagged values have a name and a type, and are associated to a specific stereotype. Graphically, tagged values are specified as attributes of the class that defines the stereotype.

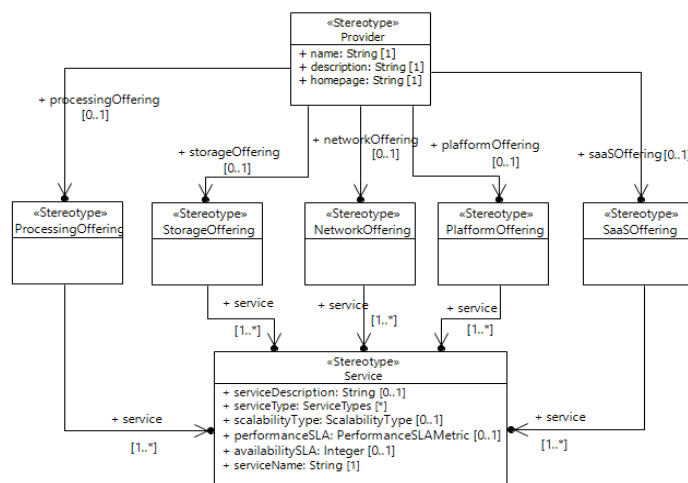


Figure 11: Example for stereotype from core profile of CloudML@artist

4.2 Key Concept 2: Model-Driven Approach to Cloud

4.2.1 Model-Driven Engineering in a Nutshell

Model-Driven Engineering (MDE) refers to the systematic use of models as primary engineering artifacts throughout the engineering lifecycle in order to raise the level of abstraction and model transformations to increase the degree of automation in the development of software [89].

Models are used to represent a certain kind of information, e.g., a model of a system that exists or that should be realized.

Transformations are the active parts that manipulate models in a systematic manner for a given purpose and generate automatically a target model from a source model according to a transformation definition which is comprised of transformation rules that describe how one or more constructs in the source language can be transformed into one or more constructs in the target language.

A transformation takes a PIM and transforms it into a PSM. In order to achieve a model transformation the appropriate tools are needed. A second or the same transformation tool transforms the PSM to code. Development tools should not only offer the possibility of applying predefined model transformations on demand, but should also offer a language that allows (advanced) users to define their own model transformations and then execute them on demand.

A transformation definition is comprised of transformation rules. Transformation rules defined for the PIM to Relational PSM transformation take care of consistent object-relational mappings. These rules describing how the elements in PIM can be mapped to elements in Relational PSM are discussed in [90]. Transformations may be also treated as models due to the model engineering principle “everything is a model”. Although, sometimes it is helpful to distinguish between models and transformations, in other case the general notion of model brings several benefits also for the development and usage of model transformations.



Figure 12: The two key concepts of MDA are models and transformations [117]

4.2.2 Model-Driven Architecture

Object Management Group (OMG) introduced the Model Driven Architecture (MDA) as an approach to software development which is characterized by the use of models as primary artifacts for understanding, design, construction, deployment, operation, maintenance and modification of a system. MDA introduces higher levels of abstraction, enabling organizations to create models that are independent of any particular technology platform. These models are simple as they describe only the essential features of the system and helps in better understanding the system. The models are described at three different levels of abstraction [91]:

- CIM (Computation Independent Model) is a software independent business domain model that bridges the gap between business experts and system experts. It describes the basic features of the system and produces a structured and coherent document of requirement specification.
- PIM (Platform Independent Model) - describes the structure, behavior and functionality of the system in a generic manner, independent of the technology that would be used for its implementation.
- PSM (Platform Specific Model) - specifies the system in terms of implementation constructs that are specific to the implementation technology. MDA allows automation of various steps in the development process and it semi-automatically generates code from the models.

A single PIM can be transformed into one or more PSMs, each PSM being specific to the technology platform on which the system would finally be implemented. A complete MDA specification consists of a definitive platform-independent model (PIM), plus one or more platform specific models (PSM), sets of interface definitions, each describing how the base model is implemented on a different middleware platform and sets of transformation definitions. The PIM depicting the structure, behaviour and functionality is modelled only once. And then, the transformation definitions enable the transformation of the PIM to one or more PSMs.

The key to the success of MDA [92] lies in automated or semi-automated model-to-model and model-to-code transformations. The transformation tool executes a transformation definition that is specified for the purpose of transforming higher-level, platform-independent business models into lower-level platform-specific models and finally into executable code. A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed into a model in the

target language. Figure 13 depicts a PIM to PSM transformation.



Figure 13: PIM to PSM Transformation [91]

4.2.3 Model-Driven Modernization

In Model-Driven (Software) Modernization (MDM) [118], models representing legacy software are (i) (semi-)automatically discovered in a reverse engineering step and (ii) transformed until the new software satisfies the modernization requirements in a forward engineering step. This process is also currently subject to standardization at the OMG under the Architecture Driven Modernization (ADM) umbrella [93]. MDE allows automating the various steps involved in the software migration, notably reverse engineering of legacy software and forward engineering towards cloud environments.

In Model-Driven Software Modernization, the legacy software system is firstly inspected by applying Reverse Engineering techniques to provide Platform Independent Models (PIMs) that represent different relevant views of that legacy system. Forward engineering techniques are then applied to these models in order to generate Platform Specific Models (PSMs), which describes the target system and desired requirements to be fulfilled. Finally, the software artefacts composing the target system are generated from such PSM models.

During the forward reengineering phase, the generation of PSM models usually is driven by platform dependent information (e.g., when a platform allows several ways of representing a concept), which becomes Platform Dependent Models (PDMs). Thus, when applying forward engineering techniques, it is required to specify the PDM meta-models describing the particular features of a target environment. In Cloud computing case, the target is a Cloud environment which, according to the kind of services it offers, is classified into (i) Infrastructure as a Service (IaaS), (ii) Platform as a Service (PaaS) and (iii) Software as a Service (SaaS). For instance, PDM metamodels for Cloud should describe entities such as services (i.e., infrastructure, platform), tools (i.e., hardware and software components), pricing policies, ratings and other factors of Cloud offerings and providers. This section surveys the exist-

ing research and industrial approaches to describe the target environment in the context of Model Driven Engineering (MDE).

4.2.4 MDA and UML Profiles

UML Profiles can play a particularly important role in describing the platform model and the transformation rules between models. The usage of UML Profiles to specify the model of a specific platform, guarantees that the derived models will be consistent with UML. The key to a successful application of MDA is to use standard models and standard UML Profiles for implementation languages or platforms when is possible. UML Profiles for some well-known component platforms are currently available.

The most important process is the mapping between each element of the PIM, and the stereotypes, constraints and tagged values that make up the platform Profile. In order to be achieved the stereotypes of a platform Profile to “mark” the elements of a PIM and produce the corresponding PSM, already expressed in terms of the elements appearing in the target platform. A mark represents a concept in the PSM, and is applied to an element of the PIM to indicate how it must be transformed into the target PSM.

4.2.5 Available Standards/Profiles/MetaModels of the chosen technology

This section studies available initiatives that provide formal and standardized meta-modelling descriptions of Cloud environments, in particular those which are technically compatible with the MDE baseline, although other Cloud meta-modelling technologies that could complement the former ones will be considered. In particular this section details existing standard Cloud meta-models as well as other meta-models suggested by on-going research. This section concludes with a comparison of modelling features among considered meta-models and a deep analysis of needed extensions.

4.2.5.1 REMICS PIM4Cloud and CloudML

Description

As its descriptive name says, PIM4Cloud is a Platform Independent Model oriented to cloud infrastructures. This means it allows the specification of cloud related issues from a platform independent model point of view, so

the resulting model could be implemented in different cloud providers now and in the future when new needs or capabilities may arise.

In short, PIM4Cloud [94] is a metamodel and UML profile for describing IT systems deployments to cloud platforms from an application designer perspective. PIM4Cloud proposes a domain-specific language (DSL) [95] to support application deployment to cloud platforms. Using this DSL, applications can be modeled in terms of components, their properties and connections (topology). Similarly, the cloud provider offerings can be described on a component level. To fulfill the requirements of the modeled application, the PIM4Cloud interpreter matches the description of the application to the description of the platform. It then deploys the application, returning a “living model” of the application, which is annotated with run-time properties. The PIM4Cloud interpreters are implemented in Scala.

The provisioning of cloud resources required to deploy and run an application is addressed by REMICS CloudML [94]. The approach proposes a common model to represent nodes that captures both design-time properties (e.g., memory, core, disk, and location) and runtime properties (e.g., public and private IP address).

Research background and motivation

In the scope of REMICS Project (<http://www.remics.eu/>), in which are involved companies closely related to the OMG was detected a lack of standards to describe cloud deployment scenarios.

To solve this shortcoming, the analysis of the state of the art prior to the creation of PIM4Cloud specification, was focused on the study of various existing standards in order to identify cloud related elements and evaluate if they were subject for modeling.

After this study, a number of concepts emerged as a result of deployment modeling study of existing approaches. These concepts served as a reference for creating PIM4Cloud.

It was three standards from which these concepts were extracted: UML 2.3, Amazon Cloud Formation and TOAF.

Cloud providers are fully aware of the importance of providing adequate support for the end customers to deploy their applications on providers’ infrastructure. That is why there have appeared so different APIs, both proprietary and open to try to abstract the characteristics of different providers in order to facilitate and automate as much as possible the deployments.

However, there was still a huge gap in terms of abstraction and modeling of the characteristics of different IaaS providers and so it was decided to approach the solution to this need through PIM4Cloud.

Target

PIM4Cloud was created to support the description and deployment of cloud applications. One of the goals taken into account when developing this specification was that it should serve to support automatic deployments in the cloud. The PSM (Platform Specific Models) generated from PIM4Cloud should serve as a starting point for the generation of specific parameters and structures to deploy an application in a particular infrastructure.

However, this requirement was not completely covered in the scope of REMICs project, and thus represents a challenge to achieve in this project.

Main characteristics

PIM4Cloud standard is aimed primarily at designers and application deployment responsible working together to develop and deploy an application to the cloud.

However, it is important to note that the use of PIM4Cloud can meet only the needs of modeling IaaS providers, as PIM4Cloud does not provide elements to abstract PaaS, due to a decision taken in relation to the high degree of heterogeneity and the fast evolution in the PAAS domain. PIM4Cloud was planned to be implemented as a profile for UML.

UML can describe most of the needs of a system that will be deployed in a cloud environment, such as which components will be deployed (components diagrams) and how these components relate (class diagrams and interaction diagrams).

Nevertheless, UML has some limitations that were tried to be addressed in the context of PIM4Cloud.

The two shortcomings of UML for modeling cloud systems addressed were:

- Limitations on modeling service architectures. This aspect was already addressed at the time of the creation of the SoaML UML extension, standard that was reused when creating PIM4Cloud.
- Limitations on shaping the deployment of components in cloud environments. The possibilities offered by UML in this regard are too general and did not allow specifying aspects directly related to the cloud.

Below are two tables: a table with the stereotypes that make up PIM4Cloud (**Error! Reference source not found.**), whose combinations based on differ-

ent diagrams (**Error! Reference source not found.**) permit to model a complete cloud infrastructure.

Extension capability

PIM4Cloud was created as an extension of UML, which is a standardized mechanism through which it can extend the UML capacity to adapt to new needs.

For this reason, in principle there should be no problems in extending PIM4Cloud to model new environments and specific characteristics of them as they arise.

4.2.5.2 CloudML: Cloud Modeling Language

The Cloud Modeling Language (CloudML) [96] is an XML-based approach that addresses service, resource, and request descriptions from an infrastructure perspective. Offerings of cloud providers are modelled in terms of services. A service is specified with a profile-based approach. A profile refers either to nodes or links and their characteristics in terms of offered CPU, storage, memory and operating system, and delay and rate, respectively. Additionally, the locations (country, state, and city), where these services are available, can be defined. Node profiles and link profiles are aggregated in terms of service types.

Second, CloudML supports the representation of all physical and virtual cloud resources, including their current state. As a consequence, this part of the language consists of two sub-parts. The first sub-part specifies XML Schemas to describe infrastructure resources. Both physical and virtual nodes can be defined. They can be connected by physical and virtual links, respectively. The second sub-part defines XML Schemas to capture the state of physical and virtual nodes. The former includes node characteristics as supported by the service description, whereas the latter additionally captures the state of virtual machine, i.e., stopped, running, and suspended.

Finally, CloudML allows service requests to be specified and related to concrete services offered by a cloud provider. A request basically contains the required nodes and links according to defined service profiles and assigns them client-specific identifiers.



























Image	Type	Description
	CloudApplication	Application whose architecture is based on the use of a cloud computing platform.
	CloudProvider	Service provider of a platform for cloud computing.
	PublicCloud	Public cloud computing platform, provided by an external provider.
	PrivateCloud	Private cloud computing platform installed and managed directly by the end user.
	CloudResourceSet	Collection of resources provided by a cloud computing platform.
	CloudResource	Generic resource provided by a cloud computing platform.
	ProcessingResource	Processing resource provided by a cloud computing platform.
	StorageResource	Storage resource provided by a cloud computing platform.
	ProcessingResourceType	Configuration of a processing resource.
	VirtualImage	Virtual disk image used by a cloud computing platform as a given configuration of a calculation resource.
	DeploymentConfiguration	Configuration of deployment of a calculation resource.
	OperatingSystem	Operating system installed on a virtual disk image.
	SoftwareComponent	Component forming the software layer of application deployed on the cloud.
	ApplicationFramework	Software component acting as an intermediary between the operating system and application components.
	ApplicationComponent	Software component that implements business aspects of the application.
	Infrastructure	Physical infrastructure of applications and private cloud computing.
	NetworkNode	Network node of physical infrastructure.
	NetworkConnection	Connection between various elements of physical infrastructure.
	Server	Component of physical infrastructure acting as a server.
	WorkStation	Component of physical infrastructure acting as an end-user of a service application.
	Network	Component of the physical architecture acting as a public or private network.
	Router	Component of physical infrastructure acting as a router.
	Switch	Component of physical infrastructure acting as a switch.
	Bus	Component of physical infrastructure acting as a data bus.
	ResourceSet	Set of generic resources.
	Resource	Generic resource.
	ResourceType	Type of generic resource.
	Property	Resource terminal.

Table 4: PIM4CLOUD stereotypes table [96]







Image	Diagram Type	Description
	ApplicationCommunication diagram	Modelling of communication and dependencies between different elements of applications.
	SoftwareComponentDependency diagram	Modelling of software architecture deployed on a virtual image disk.
	SoftwareDeployment diagram	Modelling deployment of application components and dependencies between them.
	CloudConfiguration diagram	Modelling configuration of different resources provided by a cloud computing platform.
	ResourceAssembly diagram	Modelling resources of a cloud computing platform used when deploying the application.
	PhysicalResource diagram	Modelling physical architecture of application and a private cloud computing platform.

Table 5:PIM4CLOUD diagrams reference table [96]

4.2.5.3 SysML

Description

UML (Unified Modeling Language) is a general purpose modeling language, but it has a bias for object-oriented software systems. However, in UML 2 additional modeling concepts have been added to target also other domains such as component-based systems and business processes. When dealing with systems involving software and hardware, these systems are a mixture of discrete and continuous subsystems. The software systems are discrete, while the hardware systems are continuous.

The need to model also such systems triggered the development of SysML (Systems Modeling Language) that is designed as a UML profile and already supported by several modeling and analysis tools. Especially, the later point is very important for systems design. The SysML standard has been published in its first version back in 2006 and has since then been subject to several revisions. The latest revision has been released in June, 2012 [97].

UML vs. SysML

Although, SysML is designed for modeling software/hardware systems, it may be used also to model purely software systems or software infrastructures. However both modeling languages have influenced each other. This is not surprising, because both modeling languages have been standardized by

the same institution, i.e., OMG, and they have been developed at the same time. Furthermore, the goal of SysML was to keep UML as it is as much as possible and extend it only for the needs of systems engineers that could not be covered by the UML 2 standard without developing further extensions.

As UML, SysML is designed to model the structure and the behavior of systems. Although, also for UML the goal was changed from modeling software systems to modeling systems in general, UML still does not provide many capabilities to model physical nodes except when using the deployment diagram.

Some aspects which are of great interest for systems engineers are not covered by UML. These are in particular structured textual requirements and parametric equations that are heavily used by systems engineers in practice. This need is reflected in extending the structural and behavioral modeling capabilities of UML with some specific concepts for software/hardware systems in general.

Figure 14 gives an overview on SysML and UML diagram types. The figure explicitly defines the diagram types which are supported by UML only, by SysML only, and by both modeling languages. As can be seen, both SysML and UML provide means for modeling structures and behavior of systems. While UML provides 14 different diagram types, SysML uses only 8 different diagram types. In general, when comparing the modeling concepts between UML and SysML, SysML provides less model concepts than UML. Here it has to be mentioned that SysML is described in most documents as an extension of UML and at the same time as a restriction of UML. By this, we mean that diagram types that are not explicitly mentioned by SysML, such as Communication Diagrams, Timing Diagrams, Interaction Overview Diagrams, Class Diagrams, Composite Structure Diagrams, Profile Diagram, Component Diagram, Object Diagram, and Deployment Diagram, are not part of SysML. Instead, only Sequence Diagrams, Activity Diagrams, State Diagrams, and Use Case Diagrams are roughly reused by SysML and in addition to them, SysML introduces four new types of diagrams, namely the Requirement Diagram, Block Diagram, Internal Block Diagram, and Parametric Diagram.

While the Requirement Diagram is a syntactical extension of the Class Diagram, it is used for defining cross cutting concerns. By using this diagram type, first, the requirements of a system can be defined using text in a structured way. Then, the defined requirements can be related to other requirements as well as to other modeling elements defined in different diagrams. SysML provides various relationship types such as 'derived', 'satisfied', 'verified' and 'refined'. In this way, a hierarchy of derived or refined requirements

can be built. Furthermore, those systems elements that satisfy or verify requirements can be related to them.

Requirements are classified in SysML as cross-cutting concern modeling constructs and are thus presented in the diagram type taxonomy between the behavior diagrams and the structure diagrams.

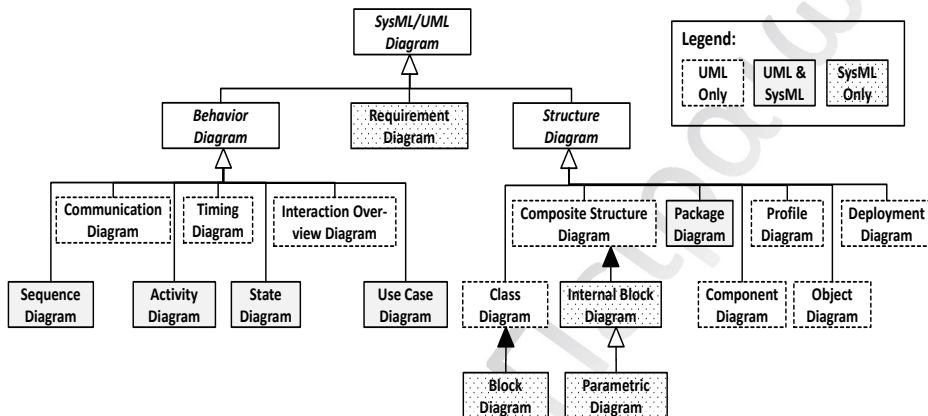


Figure 14: SysML and UML diagrams

Main characteristics

Picking interesting features of SysML is always possible, because the stereotypes of SysML may be used as annotations for already existing UML models to refine them or to allow for enhanced code generation or analysis.

Requirement diagrams from SysML may fit the purpose of defining the requirements for a migration of legacy software to cloud-based software. For instance, requirements may be not only defined for software/hardware systems, but just for software systems as mentioned in [98].

Parametric diagrams may be used to explore the performance behavior of Cloud applications by simulation. In **Error! Reference source not found.**, we show an example for parametric diagrams taken from [99]. As one can see in the Figure, an equation for defining an objective measurement function for determining the cost effectiveness of a system is defined.

To summarize, SysML may be applied to model the architecture and the components of cloud-based infrastructures by using block diagrams and to define specific configurations of cloud environments for given cloud-based software. Furthermore, the requirement diagrams in combination with par-

ametric diagrams may be employed to capture the requirements that should be fulfilled by an application and state how they have to be satisfied and verified. Parametric diagrams may act as a technique to calibrate the Cloud environment settings for given applications. However, for performing such calibration, external tools, offering analysis support, have to be employed. In this respect, the question arise which language to employ for defining the equations. This is left open by the SysML standard, but there are concrete pointers given to MathML[100] or OCL[101].

Extension capability (Metamodel and model level)

SysML is designed as a UML Profile, thus additional concepts may be integrated for cloud specific environments.

Furthermore, mixing SysML with other UML profiles such as PIM4Cloud seems to be a possible way to go. However, if several profiles are used at once, additional constraints may have to be defined or a composition of the code generators may be necessary. One alternative would be to have different profile applications for the same model that are used for different tasks. For instance, the PIM4Cloud profile may be used for code generation, while the SysML profile may be used for analysis purposes. Still, it has to be decided if a combination of the profiles, resulting in one profile application model, is useful, or if the profile applications should be separated from each other.

4.2.5.3 fUML

Description

The standard called “Semantics of a foundational subset for executable UML models” or foundational UML (fUML) [102] **Error! Reference source not found.** in short is a standard of the Object Management Group (OMG) which was released in February 2011. It formally defines the semantics of a selected subset of UML 2.3 which is called foundational UML. In essence, the fUML standard defines a virtual machine capable of executing UML models which comply with the fUML subset. This subset comprises parts of UML class diagrams as well as UML activity diagrams. I.e., using the virtual machine defined in the fUML standard, UML activity diagrams can be executed.

Research background and motivation

UML is the most adopted modeling language in industry. However, one major point of critique regarding UML is that it is lacking a precisely and com-

pletely specified semantics. The semantics of UML is informally defined in English prose and this definition is scattered throughout the standard comprising 1.000 pages. This leads to ambiguities regarding the semantics of UML models and therefore to diverging interpretations of UML models. This also led to the development of tools for executing UML models that are not interoperable because they implement different execution semantics.

Target

To overcome the limitation of UML regarding its imprecise and incomplete specification of semantics, the OMG elaborated and released the fUML standard, which contributes a formal definition of the operational semantics of a key subset of UML 2 in terms of a well-defined virtual machine for executing UML models. This subset consists of key parts of UML class diagrams and UML activity diagrams.

With the introduction of the fUML standard, UML evolved from a descriptive language that can only be used for informal design sketching to a prescriptive language that can also be used as a programming language [103].

fUML enables the simulation and execution of UML activity diagrams. Simulating a model can help in getting a better understanding about the modeled system, in ensuring the quality of models and it enables the analysis and verification of models based on the formal specification of the execution semantics of fUML models. Executing a UML model means that UML models are becoming the actual implementation of a software system, instead of only serving as the specification of the software system which is used as input for programming.

Although the semantics specification of UML provided in the fUML standard is a major step towards the utilization of executable UML models, the full potential of UML model execution cannot be exploited due to several factors. First, the standardized virtual machine lacks in providing the means for runtime observation, analysis, and execution control. Moreover, it is currently unclear how the runtime information of executable UML models can be obtained from the virtual machine and how it may be represented adequately in terms of a runtime model. As a result, important applications of models at runtime, such as controlling, observing, and adapting the behavior of a system at runtime, cannot be realized using fUML so far. Second, fUML enables the execution of UML activity diagrams. Other behavioral models of UML such as state machine diagrams and sequence diagrams as well as executable domain-specific modeling languages are not supported.

The aim of the research project moliz [104] carried out by Vienna University of Technology is to overcome these limitations. The first limitation is tackled

by the elaboration of a trace model for fUML, which enables the runtime analysis of executed UML models establishing the basis for runtime adaptation. Furthermore, an event model and a command API are developed, which enable to observe and control the model execution process during runtime **Error! Reference source not found.** The second limitation of fUML is addressed by developing an operational semantics approach using fUML for specifying the behavioral semantics of modeling languages [105].

Main characteristics

fUML specifies the semantics of UML class diagrams as well as UML activity diagrams and provides a virtual machine capable of executing activity diagrams. Activity diagrams can be used to describe the behavior of a system. It is concerned with the description of the steps necessary to accomplish a given task. An activity diagram can be used to describe workflows at a very high level of abstraction as well as to describe the instructions necessary in an operation of a class at a very low level of abstraction. An activity diagram might for instance be used for describing procedural computations, business processes, workflows, information systems, and system level processes.

A fUML activity can consist of the following modeling concepts.

Actions. An action represents a single step within an activity. fUML supports the following types of actions.

- **Object actions** for creating, destroying, modifying, and querying objects
- **Link actions** for creating, destroying, modifying, and querying links between objects
- **Communication actions** for enabling the synchronous or asynchronous communication between different activities

Control nodes. Control nodes can be used to coordinate the execution of actions in an activity. The following types of control nodes are supported by fUML.

- **Initial nodes** for defining the starting point of an activity
- **Activity final nodes** for determining the end of an activity
- **Decision nodes** for defining alternative execution branches
- **Merge nodes** for merging alternative branches again
- **Fork node** for modeling concurrent execution branches
- **Join nodes** for synchronizing concurrent execution branches

Structured activity nodes. Structured nodes can be used to group parts of an activity. The following special types of structured nodes are included in fUML.

- **Conditional nodes** for grouping parts of an activity that shall be executed if a specific condition is fulfilled
- **Loop nodes** for looping over a part of an activity multiple times
- **Expansion regions** for looping over a collection of values

Control flow. The control flow defines the flow of control through the nodes of an activity.

Object flow. The object flow specifies the flow of data through the nodes of an activity.

In the context of describing IaaS and PaaS providers required in the ARTIST project, fUML models can be used wherever a procedure has to be described, for instance for describing deployment procedures or the communication between different elements of an application. By executing these models using the fUML virtual machine, those behavior descriptions can be further analyzed.

Extension capability

fUML supports the execution of UML activity diagrams complied to the UML subset which is contained in fUML. This subset comprises a selected set of modeling concepts as described above. However, fUML could be extended in order to support additional modeling concepts. The following two possibilities exist for doing so.

(1) In fUML, the visitor design pattern is used to define the execution semantics of each supported modeling concept. I.e., the semantics of each modeling concept is specified by the implementation of a visitor class. Therefore, additional modeling concepts could be added to fUML by first adding the modeling concept in the metamodel of fUML and second implementing a visitor class specifying how an instance of the modeling concept is executed.

(2) The fUML standard defines a so-called foundational model library which is intended to serve as a library for user-level model elements which can be used in fUML models.

4.2.5.4 Cloud4SOA semantic model

Description

The Cloud4SOA semantic model [106] is an ontology designed to enable PaaS semantic compatibility and interoperability among the different and usually incompatible PaaS offerings, through the Cloud4SOA platform. Cloud4SOA semantic model defines a vocabulary, in the context of a Cloud Platform as a Service (PaaS), for expressing concepts or entities and their relationships, concerning both developer applications and PaaS offerings from different providers.

Research background and motivation

Cloud4SOA (<http://www.cloud4soa.eu/>) provides an open semantic interoperable framework for PaaS developers and providers, aiming at addressing the current problem of lack of interoperability among existing OSS or commercial PaaS offerings. This lack of interoperability comes to scene in terms of both conceptual and technical (API-level) incompatibility.

The Cloud4SOA system supports developers of Cloud-based applications with multiplatform matchmaking, deployment, service governance, monitoring or migration, by semantically interconnecting heterogeneous PaaS offerings across different providers that offer compatible technology.

Semantics play a catalytic role in the whole process of Cloud4SOA, where they are used for annotating Cloud resources or services and developer applications, expecting to significantly address their semantic interoperability. Moreover, semantics facilitates the matching between applications and those platform resources they require. Semantically annotating Cloud resources also allow to easily identify clusters of collaborating and/or complementary resources.

Target

Cloud4SOA semantic model targets application developers and PaaS providers to express, using a common vocabulary, similar concepts (i.e. application requirements or platform offering resources), while maintaining and enforcing different platform-specific or domain-specific entities and descriptions, in order to enhance the ability of Cloud4SOA system to find matchmakings between platform offerings and developers' requests.

Cloud4SOA semantic model supports developers to describe their applications in terms of required technological constraints (i.e. compatible devel-

opment language or framework, QoS attributes, service dependencies, storage needs, computational resources, etc).

Cloud4SOA semantic model supports PaaS providers to define their offerings in terms of their offered hardware and software resources, such as their computation, bandwidth, memory or storage capabilities, or their offered services, i.e. SQL or Non-SQL storage services, etc.

Main characteristics

Cloud4SOA semantic model has been engineering using the METHONTOLOGY [107] methodology and the “meet-in-the-middle” approach, where the model is the jointly result of two complementary approaches:

- A top-down that exploits existing ontologies, whose most general concepts are reused and new more specialized ones are derived from them. This methodology has been applied for applications, deriving concepts from The Open Group SOA Ontology [108], Essential Meta-Model [109] and TOGAF 9 Meta-Model [110].
- A bottom-up that defines generic ontology concepts from domain specific concepts and their relationships obtained from the survey analysis of state of the art for many PaaS platforms. This approach was adopted for PaaS providers since no standard Cloud ontology is available.

Cloud4SOA semantic model is structured into 5 ontology layers:

- The Infrastructure layer contains definitions for concepts used to capture knowledge related to the infrastructure (hardware and software) utilized by the Platform and Application layers, as well as metrics to measure the values of hardware/software attributes. Hardware and software resources are classified by categories. Examples of hardware categories are: network, storage or processing. In case of processing, Cloud4SOA semantic model defines equivalencies between different processing types and their measurement units.
- The Platform layer contains definitions for concepts used to capture knowledge related to a Cloud-based platform (e.g. supported programming language, offered software/hardware functionalities, offered APIs for programmatic access and supported communication channels, pricing policies, ratings, SLA, etc.). The platform relies on the Infrastructure layer in order to operate.

- The Application layer contains definitions for concepts used to capture knowledge related to a Cloud-based Application during the whole application Cloud engineering cycle, such the application description, its deployment description, its status after deployment, etc. A Cloud-based Application is developed/deployed/managed in a Cloud Platform.
- The Enterprise layer contains definitions for concepts used to capture knowledge related to the enterprises involved as Cloud suppliers (e.g. PaaS providers, IaaS providers, service providers, software providers, etc.) and their role in the Cloud.
- The User layer contains definitions for concepts used to capture knowledge related to the users of a Cloud platform, such as the Cloud-based application developers and the Cloud PaaS providers.
- In support of the five main layers of Cloud4SOA ontology model, some classes have been developed to represent all the metrics concepts that could be involved in modeling applications and offerings, to measure the values of hardware/software attributes

Cloud4SOA ontology model can used to specify Cloud requirements on legacy applications, since was specifically designed to enable the matchmaking of application requirements to Cloud offerings. Even if Cloud4SOA ontology model only addresses Cloud platforms (PaaS) as targets, it also supports the specification of their underlying Cloud infrastructures (IaaS) features.

Extension capability

Ontology engineering methodologies and technologies encourage on the reuse and extension of existing ontologies when defining new ones. As commented before, Cloud4SOA ontology model was built (using the top-down approach) as an extension upon a set of existing generic and domain specific ontologies. In this sense, Cloud4SOA ontology model can be easily extended to incorporate new concerns or reuse, for instance by applying Cloud4SOA ontology model concepts in the extension of other metamodels, as those described above in this section.

4.2.5.5 Blueprint Template

According to [111] Blueprint Template provides a uniform description format for cloud service offerings which cross different computing layers. Application developers through this template have the ability to choose offerings

from multiple software, platform, infrastructure service providers and finally to customize and to compose desirable Service-Based Applications (SBAs). Two main issues which are related to the migration of SBAs to cloud are the following: The first problem is the issue of multi-tenancy of the cloud services. Secondly the difficulties which arise from the creation of SBAs and the integration with other cloud service offerings, because of the inconsistency of cloud resource descriptions and interfaces. However Blueprint template allows the flexible design and deployment of cloud services because it provides a common structure, syntax and semantics. Blueprint is divided into template sections and each of them has a set of proposed properties. Moreover the template is extensible which means that if more properties are needed they can be added. The template sections of the Blueprint that are of interest to the modelling of target platforms are the following:

- **Offering section** which includes: **capability**(should be described in such a way that consumer can understand and query it from a blueprint repository),**service signature** which describes information for functionality of the offering, **functionality** and **API location** for downloading APIs, **endpoint location** for programmatic interactions with the cloud service, **Range Nr Of Instances** which includes the minimum and maximum number of instances of the cloud service provided to consumers, **QoS Profile** which includes QoS properties of the cloud service in a number of separate profiles using add-on templates or external languages and finally the **policy profile** which includes the policy rules that constrain the cloud service offering.
- **Resource Requirements Section:** Each resource requirement is specified with a resource ID, the required functionality, the required Range Number of Instances, and a set of references pointing to QoS Profiles and Policy Profiles that contain the QoS properties and the policy constraints of this resource requirement.
- **Virtual Architecture Topology (VAT) Section:** The VAT section specifies Requirement relationships: This relationship indicates a deployment dependency between two elements, e.g. an implementation artefact needs a required resource for its deployment.
- **Invariants Section:** The blueprint provider can specify the resource constraints that prescribe the conditions for all the cloud resources needed for the blueprint, as well as the QoS and policy constraints in separate QoS Inv Profiles and Policy Inv Profiles respectively.

4.2.5.6 CloudML@artist

CloudML@artist is a meta-model [119] that specifies all the concepts and relationships of interest when modelling a cloud provider. The meta-model defines the concepts and relationships that describe the main capabilities of resources offered by cloud platforms and it is realized as an extension to the UML meta-model. Therefore, it has been realized in terms of a profile / collection of profiles regarding specific aspects such as Availability and Performance Concepts. Profiles created starting from this meta-model can be used during the migration of an application in order to select the target platform that matches best the requirements and functionalities needed by the re-engineered application.

The CloudML@artist meta-model contains some concepts not covered by the original CloudML; in particular:

- PaaS and SaaS offering: the original CloudML only focuses on resources at IaaS level.
- performance and monitoring aspects: they are not taken in consideration in the CloudML meta-model, but needed in case of making decisions about the best target environment for the migration and the evaluation of the effectiveness of the migration.
- other aspects like pricing, scalability, availability, regarding the non-technical evaluation of the migration process (e.g., the business feasibility).

Chapter 5

CloudML@artist Implementation

In the previous chapter a set of meta-models solutions was presented. The most appropriate meta-model from the aforementioned ones that could extensively describe Cloud Providers from a performance point of view and could be extended in order to create the respective instances for specific cloud providers, is the CloudML@artist meta-model.

The main purpose of this thesis, as was mentioned in Thesis Statement section, is the extension of CloudML@artist meta-model in order to directly insert performance information of Cloud offerings in a variety of application types, through the use of relevant benchmarks. To accomplish this, apart from the creation of specific instances for each Cloud Provider by using CloudML@artist meta-model, a set of tools that are presented in this chapter is needed.

5.1 Why CloudML@artist

In this section the most important reasons that lead to CloudML@artist selection are included. The main reason is that CloudML@artist meta-model extends the CloudML definition in order to cover aspects regarding performance and availability and also provides a better description of service offerings on different levels (PaaS and SaaS).

Moreover, it covers the lack of adequate description frameworks for capturing performance characteristics of cloud services and resources. For example, for CPU resources typical descriptions (like in CloudML) include only number and frequency of CPU cores. However this is far from sufficient for accurately describing the actual performance of a computing resource. Furthermore, fluctuation in the actual output of these services due to cloud environment issues (e.g. noisy neighbour effect, multi-tenancy, migration) is a severe aspect that has begun to take notice in the cloud users.

Finally, includes UML profiles for the most popular cloud providers such as Amazon EC2, Windows Azure and Google App Engine and provides the ability

to be extended by creating the respective instances that include concrete values for each of them.

5.2 UML Profiles Description and meta-model Structure

As described in previous section, CloudML@artist is organized as a set of UML profiles with hierarchical relationships between them. Next a brief description of each profile:

Core profile

UML Profile containing generic stereotypes and data types that can be applied to characterize entities belonging to different cloud providers. As can be seen in figure 4, the Core profile is divided into 3 sub-profiles for a better understandability and usage:

- IaaS: contains specific IaaS stereotypes and data types. As it is a sub-profile contained in the Core Profile, the stereotypes contained in it can extend directly those stereotypes (common stereotypes) defined at a higher level and can also make use of the common data types at that level. This is applicable also to PaaS and SaaS subprofiles.
- PaaS: contains specific PaaS stereotypes and data types.
- SaaS: contains specific SaaS stereotypes and data types.

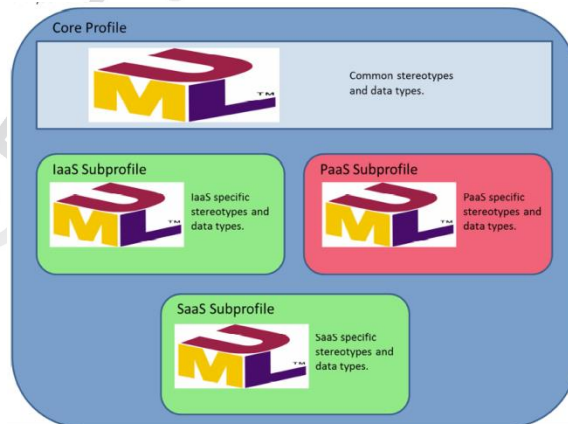


Figure 15: Core Profile

Amazon EC2 profile

Such profile describes Amazon EC2 provider and allows creating models to specify values for concrete deployments on this provider.

As Amazon EC2 is an IaaS provider, this profile imports IaaS profile and makes use of generic stereotypes defined at that level in the same way any other IaaS provider could do. The use of this inheritance mechanism is very convenient in order to not repeat the creation of stereotypes that have been defined at a higher level.

Google App Engine profile

This profile has the same objective than Amazon EC2 and Windows Azure's, but it is focused on Google App Engine specification needs. The main difference is that, taking into account that GAE is a PaaS profile, it imports and makes use of PaaS stereotypes instead of IaaS ones.

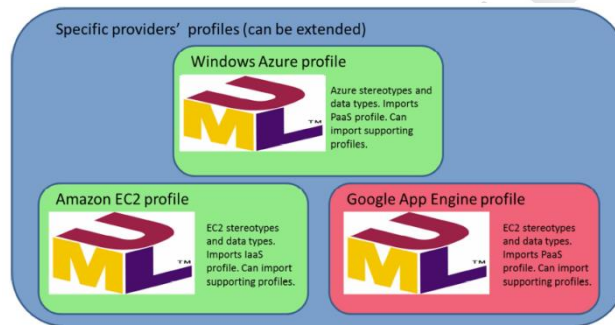


Figure 16: Specific providers' profiles

Next is described a set of "supporting profiles". These profiles have been created in order to respond to the representation needs of the project at this stage. For now the set of "supporting profiles" is composed of Pricing, Availability, Security and Benchmark profile, but it will be possible to extend it by adding other profiles in a quite simple way in case new requirements arise. Furthermore, they are independent of the CloudML@artist main profile, thus can be individually used (e.g. by other approaches).

Pricing profile

Included in "supporting profiles" category, this profile can be applied to any cloud provider to model pricing related aspects.

Availability profile

Profile that permits to model cloud provider availability related aspects, as these are expressed in the SLAs. The stereotypes of this profile can be applied on different service elements (e.g. ServiceOfferings), in order to inde-

pendently describe different SLAs that may apply to different types of services (e.g. Compute SLA, Storage SLA etc.).

Security profile

This profile is used to specify security related characteristics at provider level. At the moment the amount of modelled characteristics can be significantly enriched.

Benchmark profile

This profile can be included when modelling a cloud provider to specify results of benchmark tests, when attached to specific service instance types.

Next figure describes how **Cloudml@artist** is structured as a series of interconnected UML profiles, making possible to create models with great flexibility.

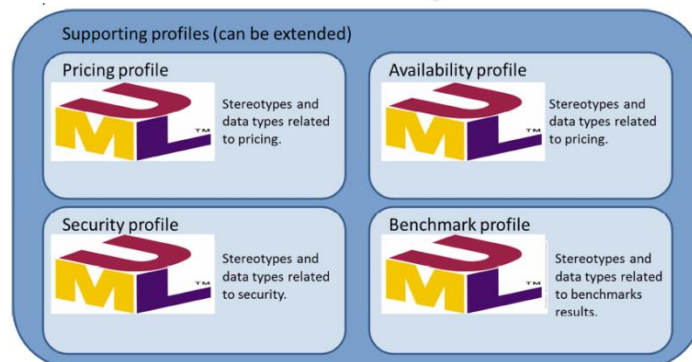


Figure 17: Supporting profiles

Detailed Description of benchmark profile

Benchmark profile, included in supporting profiles, is related to performance and is the one that can be applied to Amazon and Azure and Google App Engine profiles in order to extend CloudML@artist and add respective performance values that represent different cloud services. This subprofile includes a number of different benchmarks covering the most prominent application types and providing the ability for acquiring performance score results.

Regarding performance profile analysis, the basic stereotype is `BenchmarkedElement` which includes `BenchmarkResult` (Figure 34) stereotype, as an attribute. The latter includes only one property and is related to the potential benchmark workloads. Moreover in the definition of our performance model an OCL constraint has been generated. According to this constraint, a

BenchmarkElement can be only applied to an InstanceType element. In the same way, different constraints may be defined throughout the profiles, to link them with the core profiles.

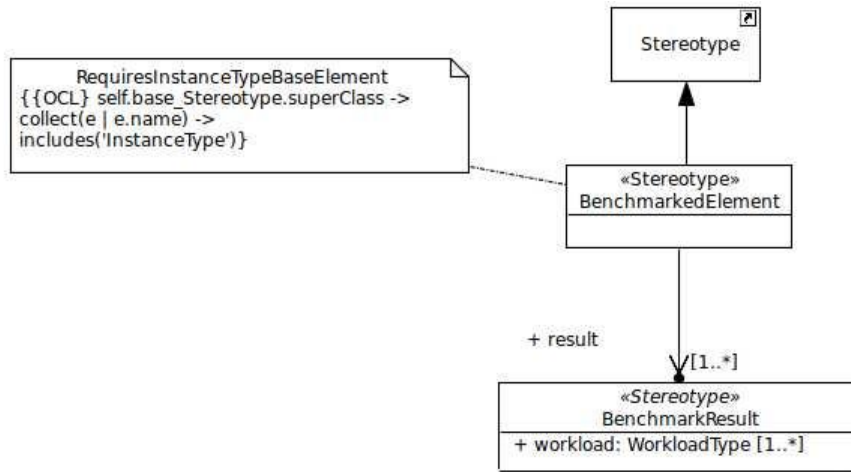


Figure 18: Illustration of the basic part of performance model and the OCL constraint

Next is described the property contained in BenchmarkResult stereotype.

Name	Type	Card.	Description
workload	WorkloadType	1..*	specific workload patterns that can be mapped to concrete applications

DacapoResult properties are described in the following table. The DaCapo benchmarks reflect performance time and are used in order to evaluate Java-based applications. Cardinality in all results can be also 0, in case no tests have been performed for this specific benchmark.

Name	Type	Card.	Description
PerformanceTime	Real	0..1	Response time for test completion

With regard to YCSBResult stereotype, YCSB reports back a number of metrics such as runtime, throughput, number of operations, average, minimum and maximum latency. These are included in the performance profile in order to describe overall results of an offering.

Name	Type	Card.	Description
runtime	Real	0..1	Execution time needed for workload completion
throughput	Real	0..1	Operations/sec
operations	Real	0..1	Update operations completed
averageLatency	Real	0..1	Average time per operations(the Client measures the end to end latency of executing a particular operation against the database)
minLatency	Real	0..1	Minimum latency
MaxLatency	Real	0..1	Maximum latency

DwarfsResult stereotype includes the problem size which is set as a real parameter and the execution time for test completion. In the following table the aforementioned parameters are summarized.

Name	Type	Card.	Description
score	Real	0..1	Runtime benchmark result
size	Real	0..1	Problem size

FilebenchResult stereotype has been defined in order to capture the typology of results, including the various statistics that are returned. In the following table the aforementioned parameters are summarized.

Name	Type	Card.	Description
ops	Real	0..1	The number of operations
throughputOpsSec	Real	0..1	Operations per second
rw	Real	0..1	Reads/writes to get a feeling for maximum performance
bandwidthMbSec	Real	0..1	Megabytes/second
cpuOp	Real	1	Number of cpu operations
Latency	Real	0..1	Latency

Regarding the Cloudsuite case, which offers a benchmark suite for emerging scale-out applications(eight application types) only the generic average score has been kept in order to be included in the model instances in order to simplify the descriptions (each of the applications reports a large number of statistics, that are case specific). Next the CloudSuiteResult stereotype is described.

Name	Type	Card.	Description
Average_score	Real	0..1	Average score related to specific benchmark application type

All of the above benchmark results, included in the performance model, are represented in Figure 19 .

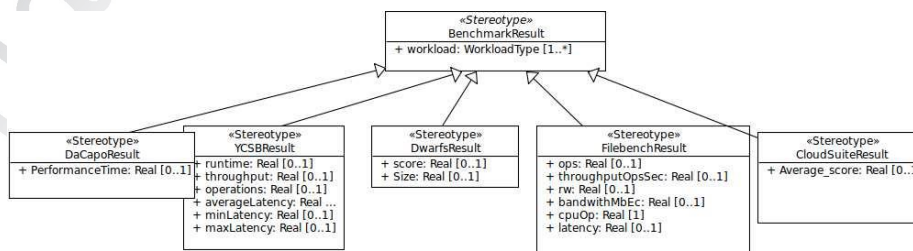


Figure 19: Benchmark results included in performance profile

In order to simplify the benchmark profile a universal enumeration has been defined (Figure 20) that includes the default workloads from the aforementioned benchmark categories. These workloads are static in order to be able to compare the performance of different services on the same examined workload.

«Enumeration» WorkloadType
YCSB_Update_Heavy
YCSB_Read_Heavy
YCSB_Read_Only
YCSB_Read_Latest
YCSB_Short_Ranges
Filebench_Webserver
Filebench_Fileserver
Filebench_Varmail
Filebench_Videoserver
Filebench_Webproxy
Filebench_OLTP
DaCapo_Avrora
DaCapo_Batik
DaCapo_Jython
DaCapo_LuIndex
DaCapo_Xalan
CloudSuite_Datacaching_Twitter
Dwarf_StructuredGrid_3DCurl
Dwarf_GraphTraversal_Quicksort
CloudSuite_MediaStreaming_GetShortHi

Figure 20: Illustration of the universal enumeration with the default workloads

Next a detailed description of the workloads included in the WorkloadType enumeration is given.

Name	Description
YCSB_Update_Heavy	a mix of 50/50 reads and writes
YCSB_Read_Heavy	a 95/5 reads/write mix
YCSB_Read_Only	100% read
YCSB_Read_Latest	new records are inserted
YCSB_Short_Ranges	short ranges of records are queried, instead of individual records

Filebench_Webserver	Emulates simple web-server I/O activity
Filebench_Fileserver	Emulates simple file-server I/O activity
Filebench_Varmail	Emulates I/O activity of a simple mail server that stores each e-mail in a separate file (/var/mail/ server)
Filebench_Videoserver	emulates a video server
Filebench_Webproxy	Emulates I/O activity of a simple web proxy server
Filebench_OLTP	A database emulator
DaCapo_Avrora	simulates a number of programs running on a grid of AVR micro-controllers
DaCapo_Batik	produces a number of Scalable Vector Graphics (SVG) images based on the unit tests in Apache Batik
DaCapo_Jython	interprets pybench Python benchmark
DaCapo_Luindex	Uses lucene to indexes a set of documents; the works of Shakespeare and the King James Bible
DaCapo_Xalan	transforms XML documents into HTML ones
CloudSuite_Datacaching_Twitter	a simulation of Twitter-type workload for in cache memory data
CloudSuite_MediaStreaming_GetShortHigh	It consists of two main components a client and a server: the client component emulates real world clients; sending requests to stress a streaming server.
Dwarf_StructuredGrid_3DCurl	Regular grids, can be automatically refined
Dwarf_GraphTraversal_Quicksort	Decision Tree, searching, quicksort

5.3 Overall Process for Instance Creation

As was mentioned in the previous chapters CloudML@artist does not provide specific performance measuring metrics for the different cloud providers. For this purpose, one of the main targets of this thesis is the creation of instances for each cloud provider. Towards this direction, each of the created instances is populated with concrete performance values providing the ability to compare and rank the different cloud services. To accomplish this, the following steps should be followed.

5.3.1 Installation of CloudML@artist

The CloudML@artist meta-model is available as a set of UML profiles compatibles with the Eclipse IDE and can be imported and used to create new models. CloudML@artist has been created by making use of Eclipse ecosystem, more in concrete by using Papyrus 0.10.1 (<http://www.eclipse.org/papyrus/>) design tool plugin installed inside Eclipse Modelling Kepler SR1, that can be downloaded freely from <http://www.eclipse.org/downloads/>. Once the design environment has been installed it is necessary to download Cloudml@artist project from the GitHub repository in order to be able to start creating models by using the UML profiles defined in the meta-model.

5.3.2 Amazon EC2 Instance creation

In this section we will go through the steps to apply performance profile to Amazon EC2. This cloud provider was selected because is one of the most popular ones and many application owners chose Amazon EC2 to deploy their components.

For this, it is needed to:

- Import Import Cloudml@artist (previously downloaded from GitHub repository) project into Eclipse. As can be seen in next screen, the meta-model is structured as a set of UML profiles stored under two folders: main_profiles and supporting_profiles.

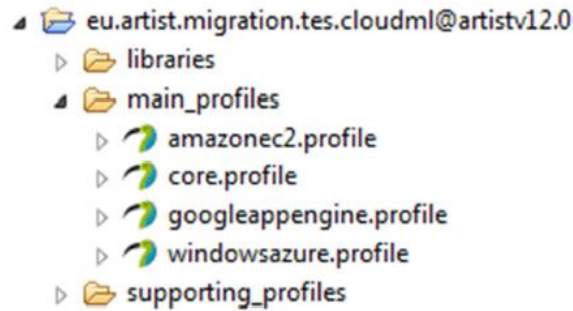


Figure 21: CloudML@artist is stored under two main folders main_profiles and supporting_profiles

- Make sure to work from the Papyrus perspective within Eclipse environment



Figure 22 :Papyrus perspective selected

- Apply the *benchmark* profile to the Amazon EC2 profile by pressing [+] button on Properties Profile tab. To be more precise BenchmarkedElement Stereotype from benchmark profile should be applied to one of the IaaSInstanceType Stereotypes. For instance, we can follow this process for M1MediumInstance Stereotype.

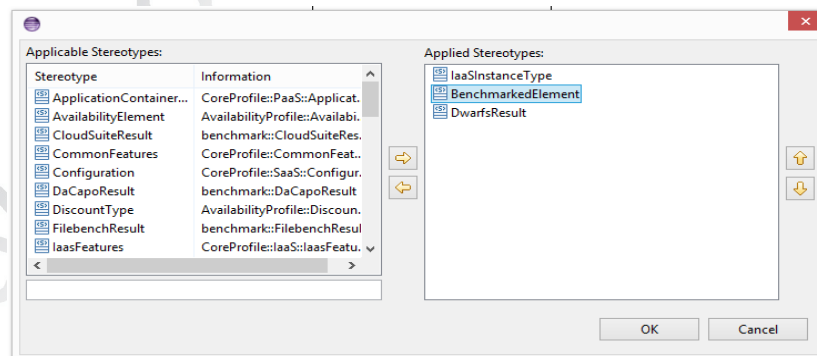


Figure 23: BenchmarkElement is applied to the Amazon EC2 profile

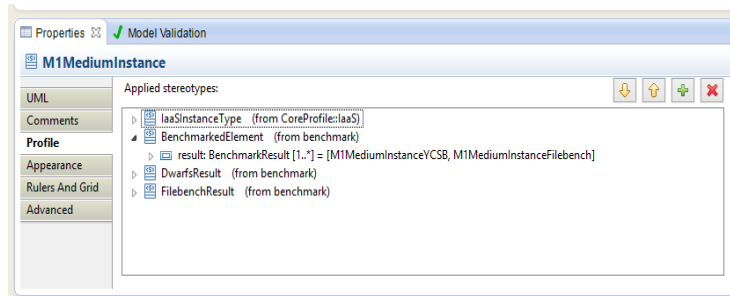


Figure 24: BenchmarkElement is incorporated in M1Medium instance

After doing this, in most of the cases there can be assigned values to the properties defined in the selected stereotype. However, for BenchmarkedElement property we notice that corresponds to one of the stereotypes that have to be defined in the profile. In this case, before applying a value to the property, first it will be necessary to include a stereotype implementing the needed stereotype (here YCSBResult, FilebenchResult or DaCapoResult). In order to create one, as can be seen in next screenshots we can drag a stereotype from the palette to the diagram. Now it is possible to assign the respective average values from benchmarking process to the M1MediumInstance stereotype.

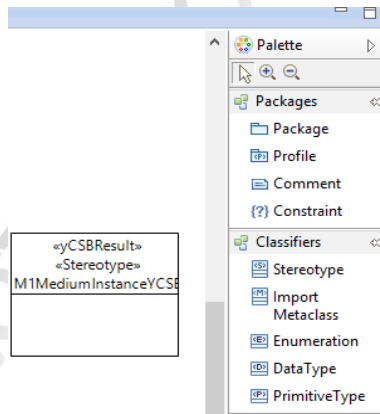


Figure 25: Implementation of a new stereotype for YCSB test results

5.3.3 Benchmarking process for performance results collection

In order to complete the Amazon EC2 instance and populate it with performance results a set of benchmarking tools is needed to be used. Moreover, in this section, the description of a MySQL raw database schema creation is included in order to store locally the results obtained by the execution of benchmark tests. Also, simple queries were created that simplifies the process of retrieving the average values for each of the different benchmarking

tests. Finally bash scripts were implemented and incorporated in the execution of benchmarking tools in order to contribute to the fully automation of the benchmarking process.

Benchmarking Controller

This tool automates the execution of benchmarking tests and eases the collection of performance data. Automation aspect is very important in our approach because by making possible to manage automatically benchmark execution saves a lot of time to users and produces better quality results.

The main objective of this component is to relieve the user from the usual work-flow of benchmarking execution that needs to be done manually: 1) creation of target environment, 2) installation of benchmarking tools, 3) execution of benchmarks and 4) retrieval of results.

The user through may set the conditions of the test, selecting the relevant benchmark, workload conditions, target provider and service offering. The Benchmarking Controller is responsible for raising the virtual resources on the target provider and executing the tests. The former is based on the incorporation of Apache LibCloud project, in order to support multiple provider frameworks. The latter needs to install first the tests, through the utilization of an external Linux-like repository that contains test executables. Once the tests are installed (through a standard repo-based installation), the workload setup scripts are transferred to the target machines and the execution begins. Results are transferred back, stored locally and processed in order to be included in the model instances.

Benchmarking tools

For the actual benchmark the set of third-party benchmarking tools that have been described extensively in chapter 3 have been selected. These are: DaCapo benchmarking suite, YCSB and Filebench. The selection of the aforementioned tools based on the following reasons a) they have been proved to work fine, b) they are supported by a large community of experts, c) there is a lot of documentation and tests already carried out and performance data already available and d) users are already familiar with them.

In this context benchmarking tools are meant to be executed several times to capture variation in the performance values. Given that Cloud users need stability in the performance of their resources benchmarking should be a repeated measurement process over time to observe variations of offering. In order to achieve the above benchmarking repetition an extensive study of the benchmarking workloads was demanded as a precondition.

Also, the configuration definition for each workload and the incorporation of a number of parameters in some cases were specified. Finally, bash scripts were implemented and used with benchmarking controller for setting the proper order for the test execution and in some cases for deleting data in the database. For instance, for YCSB the recommended sequence in order to keep the database size consistent is the following: workload A, workload B, workload C, workload F, workload D, delete data in the database, workload E.

Benchmark Database

A mysql raw database schema has been created for the locally storage of benchmarking execution. The database structure is depicted in Figure 26. Simple queries have been implemented regarding average values that may be addressed towards the backend raw data.

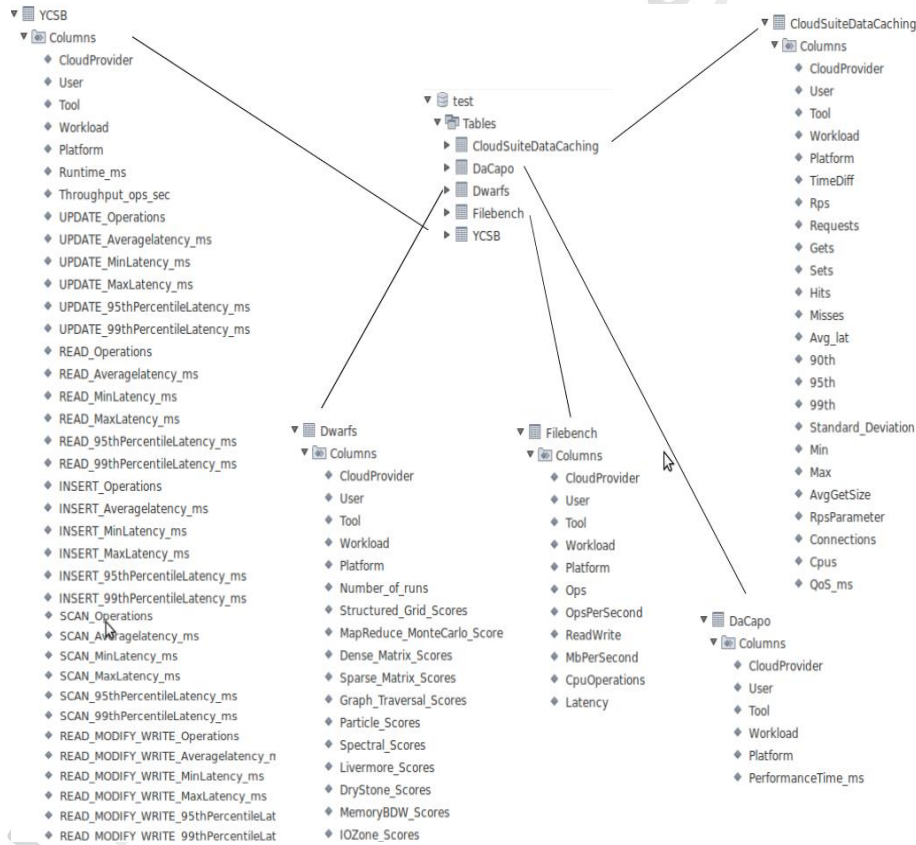


Figure 26: Mysql raw database schema

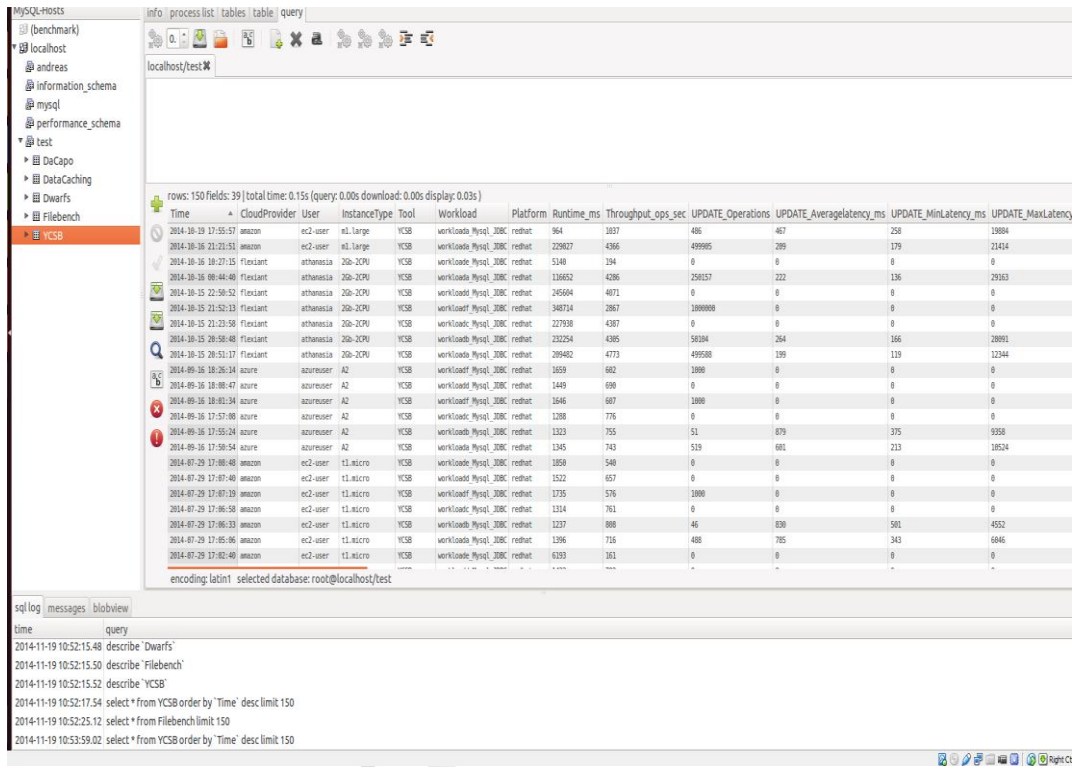


Figure 27: Local Database for storing benchmark results

CHAPTER 6

Benchmarking Case Study On Three Selected Cloud Providers: Amazon EC2, Microsoft Azure and Flexiant

In this chapter a detailed analysis for benchmarking process on three large commercial cloud providers, Amazon EC2, Microsoft Azure and Flexiant is presented. The measurement results will be included in the CloudML@artist profile in order to assist with provisioning decisions for cloud users.

6.1 Benchmarking process

In order to experiment initially with the defined metrics and investigate differences in VM performance, we utilized workloads from DaCapo benchmarking suite, YCSB benchmark framework and Filebench. However the Benchmarking Controller apart from DaCapo and YCSB supports the managing execution of two more benchmarks included in Table 1, such as Dwarfs, CloudSuite. Nevertheless, in this work, only the aforementioned ones have been tested.

During the execution process the user runs locally the benchmark controller tool specifying the target Environment to test for instance Amazon and the instance type to create and later to destroy (e.g OS, size). Also the user selects the benchmark tool to run on the remote host (e.g., DaCapo, YCSB). The scripts remote is transparent to user and the results of the execution are transferred back locally, parsed and eventually stored to the local database.

DaCapo is designed to facilitate performance analysis of Java Virtual Machines, YCSB measures databases performance, while Filebench measures file system and storage. The selected workloads from each test were running on instances in three different cloud environments: Amazon EC2, Microsoft Azure and Flexiant. Regarding Amazon EC2, different types of VM instances were selected while for Microsoft Azure and Flexiant the tests were running on the same VM instances during the entire benchmarking process. Information regarding the selected benchmarking workloads and the VM instance characteristics are presented in Table 6 and Table 7 respectively.

DaCapo	Filebench	YCSB
xalan: transforms XML documents into HTML ones	Fileserver: Emulates simple file-server I/O activity. This workload performs a sequence of creates, deletes, appends, reads, writes and attribute operations on a directory tree. 50 threads are used by default. The workload generated is somewhat similar to SPECsfs.	A: Update heavy workload
tomcat: runs a set of queries against a tomcat server retrieving and verifying the resulting webpages	Varmail: Emulates I/O activity of a simple mail server that stores each e-mail in a separate file (/var/mail/ server). The workload consists of a multi-threaded set of create-append-sync, read-append-sync, read and delete operations in a single directory. 16 threads are used by default. The workload generated is somewhat similar to Postmark but multi-threaded.	B: Read mostly workload
pmd: analyzes a set of Java classes for a range of source code problems	Videoserver: This workload emulates a video server. It has two filesets: one contains videos that are actively served, and the second one has videos that are available but currently inactive. One thread is writing new videos to replace no longer viewed videos in the passive set. Meanwhile \$nthreads threads are serving up videos from the active video fileset.	C: Read only
jython: interprets pybench Python benchmark	Webproxy:	D: Read latest workload
h2: executes a JDBC benchmark using a number of transactions against a banking model application	Webserver: Emulates simple web-server I/O activity. Produces	E: Short ranges

	a sequence of open-read-close on multiple files in a directory tree plus a log file append. 100 threads are used by default.	
fop : parses/formats XSL-FO file and generates a PDF file		F : Read-modify-write
eclipse : executes jdt performance tests for the Eclipse IDE		
avrora : simulates a number of programs running on a grid of AVR micro-controllers		

Table 6: Selected benchmarking workloads

Cloud Provider	VM instance	Region
Amazon EC2	t1.micro	N.Virginia
	m1.medium	N.Virginia
	m1.large	N.Virginia
Microsoft Azure	small Standard	Ireland
Flexiant	4GB RAM- 3CPU	Ireland

Table 7: VM instance characteristics

The execution of the tests took place at specific hours (daily and at different time intervals) during a period of two weeks and the average values were extracted for each case. Moreover, the different time zones of the three respective regions were taken into consideration so that the peak hours were the same in each zone.

6.2 Benchmarking Results

In order to draw conclusions from the execution of the benchmarks, one should compare between same color bars, indicating similar workloads. From the graphs it is evident that the performance for a specific workload varies and depends on both the type of workload and the VM instance size. For instance for DaCapo benchmark the workloads performance across Azure (A1 Standard) and Amazon(m1.medium) is almost similar apart from some cases such as tomcat and eclipse workloads where Amazon provides better results. However, for avrora workload although Amazon m1.medium VM provides more resources the performance result in Azure is significant better.

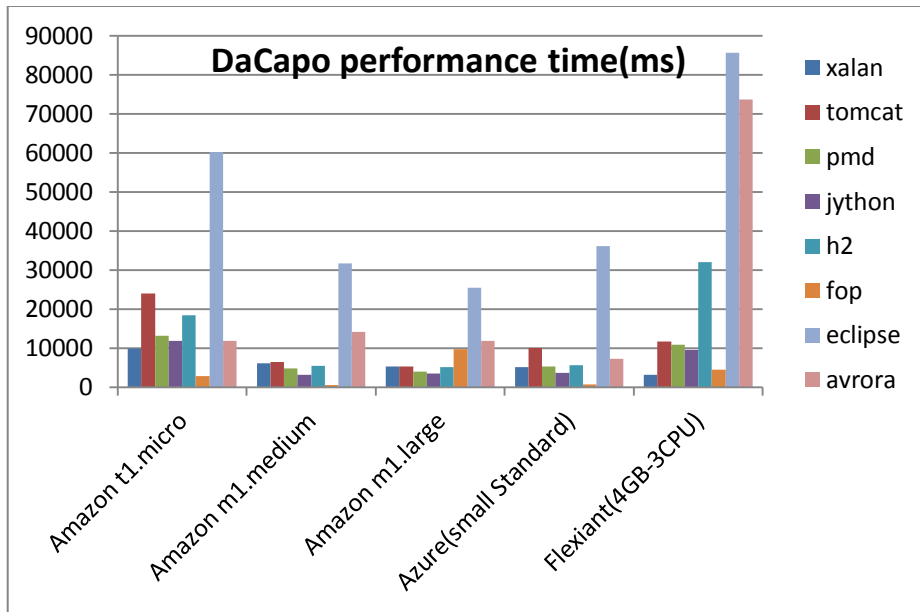


Figure 28: Performance time in ms for DaCapo workloads

Regarding YCSB, the performance for the given workloads is similar across the Amazon and Azure instances. This is probably due to the fact that the maximum computational threshold of the VM was not reached. For Flexiant the performance is significantly lower and this behaviour seems to be related to a configuration of the VM in the Flexiant environment which was outside of our control.

In addition, for all the tested VM instances the performance for the "Short Ranges" workload, 'workload_e', is approximately three times lower than the other workloads. Thus, independently from the VM size (small, medium or large) the 'workload_e' seems to be three times slower than other workloads which were tested.

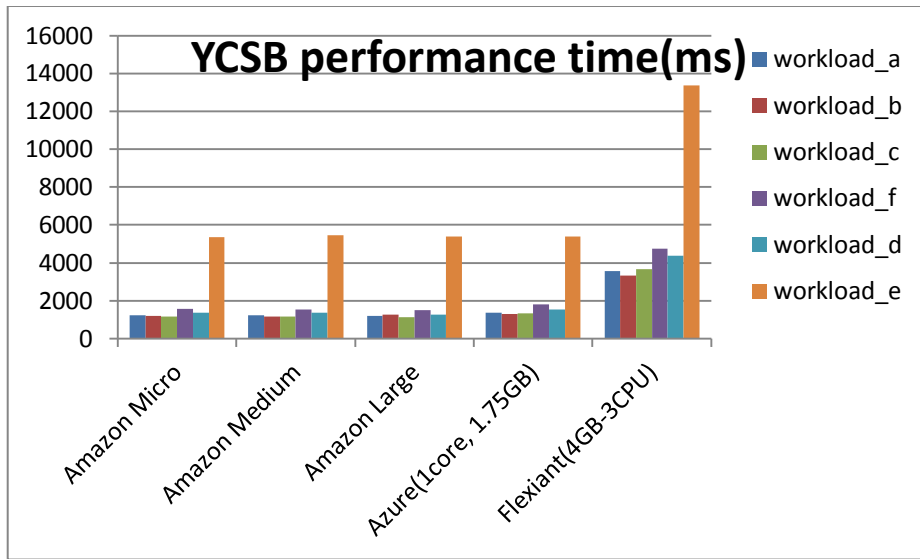


Figure 29: Performance time in ms for YCSB workloads

For Filebench the performance results are presented in figure. From the graph it is obvious that latency for fileserver and varmail workloads is significantly higher during the test execution in Azure(A1 Standard) VM. Finally, in some cases some types of VMs did not support all workloads executions as in videoserver case which could not run in Amazon t1.micro.

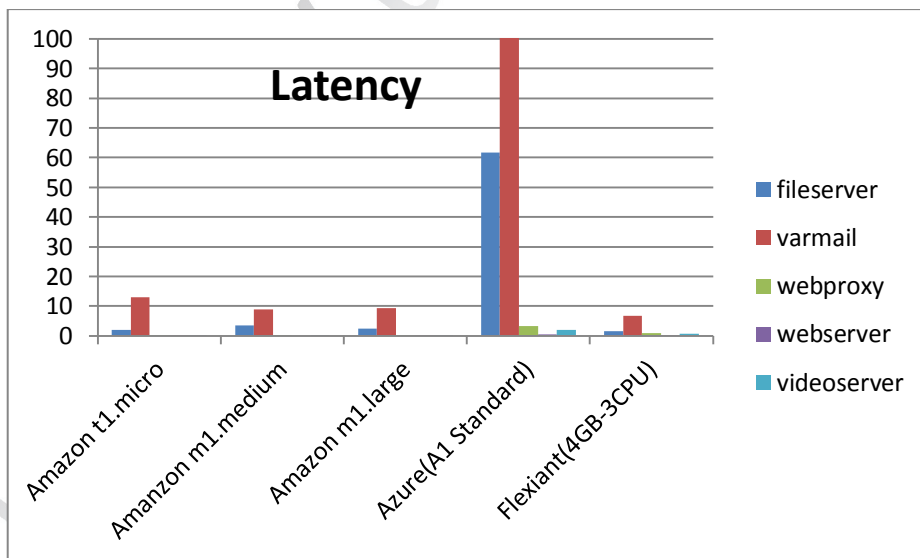


Figure 30: Latency for Filebench workloads

Chapter 7

Conclusions and Future Work

In this research work, the main objective was the extension of the CloudML@artist profile regarding performance and the implementation of Cloud Providers' instances such as Amazon, Azure and Flexiant. Performance values were included in the aforementioned instances in order to evaluate and rank the various cloud services. Benchmarking process was used as to execute tests and collect performance results, stored locally in a database.

The present work could be seen as a step towards the automated classification of an application component to a known application category. According to the previous mechanism the user after the identification of application's computational footprint will map the latter to a list of well-known computational patterns. Knowing the behavior of the application, simplifies the selection of the best cloud offering for applying the software component. For this purpose the modeling and measurement of cloud services performance is the initial and most significant contribution in order to implement the classification process. From application owner perspective, they must know in advance provider performance stability characteristics in order to achieve the best and most successful cloud migration to the Cloud (saving money and guaranteeing stability). From cloud provider perspective, the latter are able to identify the application type running in their infrastructures and enhance the management of resources. Also, as future work the measurement of service performance could be combined with the usage of specialized metrics for ranking the services according to a weighted combination of cost, performance and workload. Moreover, some other non-functional attributes could be investigated such as availability of services and could be calculated using an abstracted approach, regardless of the supported provider on which they are deployed.

Finally the classification approach could provide a solution regarding the best application combination running in cloud provider's infrastructure. Virtualization techniques are used in order to run applications with different characteristics and requirements inside Virtual Machines (VMs) on the same physical multi-core host, with increased levels of security and isolation. However, during the recent years a number of issues such as degradation of the

performance of the applications have been raised, due to the interference of different combinations of application types when running concurrently on the same node. As a future work, the research of different deployment patterns could lead to the discovery of different application combinations that show less interference and therefore overhead when running concurrently. This approach will enhance the different placement decisions and the resource management in an optimal way.

Πανεπιστήμιο Πειραιώς

References

- [1] Böhm, Markus, et al. "Cloud Computing–Outsourcing 2.0 or a new Business Model for IT Provisioning?." *Application management*. Gabler, 2011. 31-56.
- [2] Leimeister, Stefanie, et al. "The business perspective of cloud computing: actors, roles and value networks." (2010).
- [3] Kousiouris, George, et al. "Legacy Applications on the Cloud: Challenges and enablers focusing on application performance analysis and providers characteristics." *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*. Vol. 2. IEEE, 2012.
- [4] Wang, Lizhe, et al. "Scientific Cloud Computing: Early Definition and Experience." *HPCC*. Vol. 8. 2008.
- [5] Chung, Lawrence, and Sam Supakkul. "Capturing and reusing functional and non-functional requirements knowledge: a goal-object pattern approach." *Information Reuse and Integration, 2006 IEEE International Conference on*. IEEE, 2006.
- [6] Z. Zhang, X. Zhang, Realization of open cloud computing federation based on mobile agent, in: IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009, vol. 3, pp. 642–646
- [7]] M.G. Avram. —Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective||,Procedia Technology, 12, 2014, pp. 529-534.
- [8] Wang, Lizhe, et al. "Cloud computing: a perspective study." *New Generation Computing* 28.2 (2010): 137-146.
- [9] Zhang Mian, Zhang Nong; "The Study of Multimedia Data Model Technology Based on Cloud Computing"; 2010 2nd International Conference on Signal Processing Systems (ICSPS).
- [10] Marston, Sean, et al. "Cloud computing—The business perspective." *Decision Support Systems* 51.1 (2011): 176-189.
- [11] Vaquero, Luis M., et al. "A break in the clouds: towards a cloud definition." *ACM SIGCOMM Computer Communication Review* 39.1 (2008): 50-55.
- [12] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7-18.
- [13] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).
- [14] Gong, Chunye, et al. "The characteristics of cloud computing." *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*. IEEE, 2010.

- [15] Bai, Xiaoying, Jerry Zeyu Gao, and Wei-Tek Tsai. "Cloud Scalability Measurement and Testing." *Software Testing in the Cloud: Perspectives on an Emerging Discipline* (2013): 356.
- [16] Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM* 53.4 (2010): 50-58.
- [17] Kranas, P.; Anagnostopoulos, V.; Menychtas, A.; Varvarigou, T., "ElaaS: An Innovative Elasticity as a Service Framework for Dynamic Management across the Cloud Stack Layers," *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, vol., no., pp.1042,1049, 4-6 July 2012.
- [18] N. R. Herbst, S. Kounev, and R. Reussner. "Elasticity in Cloud Computing: What It Is, and What It Is Not". In: ICAC. 2013.
- [19] Agrawal, Divyakant, et al. "Database scalability, elasticity, and autonomy in the cloud." *Database Systems for Advanced Applications*. Springer Berlin Heidelberg, 2011.
- [20] Chaitanya "Virtualization Technology in Cloud Computing Environment" *International Journal of Emerging Technology and Advanced Engineering* (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 3, March 2013) pp.771-773
- [21] Gurav, U., and R. Shaikh. "Virtualization: a key feature of cloud computing." *Proceedings of the International Conference and Workshop on Emerging Trends in Technology*. ACM, 2010.
- [22] Macias, Guillermo. "Virtualization and Cloud Computing." (2013).
- [23] Singh, Aameek, Madhukar Korupolu, and Dushmanta Mohapatra. "Server-storage virtualization: integration and load balancing in data centers." *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008.
- [24] Buyya, R., T. Cortes, and H. Jin. "A Case for Redundant Arrays of Inexpensive Disks (RAID)." (2009): 2-14.
- [25] Chung, JaeWoong, et al. "Tradeoffs in transactional memory virtualization." *ACM SIGARCH Computer Architecture News*. Vol. 34. No. 5. ACM, 2006.
- [26] Data Virtualization online available at: <http://www.compositesw.com/data-virtualization/>
- [27] VMware: The Software-Defined Data Center online available at: <http://www.vmware.com/software-defined-datacenter/networking-security>
- [28] Vishwanath, Kashi Venkatesh, and Nachiappan Nagappan. "Characterizing cloud computing hardware reliability." *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010.

- [29] Shen, Zhiming, et al. "Cloudflex: elastic resource scaling for multi-tenant cloud systems." *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011.
- [30] Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM* 53.4 (2010): 50-58.
- [31] Youseff, Lamia, Maria Butrico, and Dilma Da Silva. "Toward a unified ontology of cloud computing." *Grid Computing Environments Workshop, 2008. GCE'08*. IEEE, 2008.
- [32] Parmar, Vinti, Meenakshi Chawla, and Rajender Singh. "AXIOMS OF CLOUD COMPUTING."
- [33] Hay, Brian, Kara Nance, and Matt Bishop. "Storm clouds rising: security challenges for IaaS cloud computing." *System Sciences (HICSS), 2011 44th Hawaii International Conference on*. IEEE, 2011.
- [34] A. Stanik, M. Hovestadt, and Odej Kao. Hardware as a service (haas): The completion of the cloud stack. In *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, volume 2, pages 830–835, 2012.
- [35] OpenCrowd, <http://cloudtaxonomy.opencrowd.com/>, accessed December 7, 2012.
- [36] Han, T., Sim, K.W. (2010) An Ontology-enhanced Cloud Service Discovery System, in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 (IMECS 2010)*.
- [37] Rimal, B.P., Eunmi, C., Lumb, I. (2010) A Taxonomy, Survey, and Issues of Cloud Computing EcoSystems, in *Journal of Computer Communications and Networks*, 0(0) pp. 21-46. Springer-Verlag
- [38] Höfer, C.N. and Karagiannis, G. (2011) Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications*, 2 (2). pp. 81-94
- [39] Li H, Spence C, Armstrong R, Godfrey R, Schneider R, Smith J, White R (2010) Intel cloud computing taxonomy and ecosystem analysis. *IT-Intel Brief (Cloud Computing)*
- [40] Voorsluys, William; Broberg, James; Buyya, Rajkumar (February 2011). "Introduction to Cloud Computing". In R. Buyya, J. Broberg, A.Goscinski. *Cloud Computing: Principles and Paradigms*. New York, USA: Wiley Press. pp. 1–44.
- [41] Amazon EC2 online available at: <http://aws.amazon.com/ec2/>
- [42] Microsoft Azure online available at: <http://azure.microsoft.com/el-gr/>
- [43] Google App Engine Cloud Platform online available at: <https://cloud.google.com/appengine/>
- [44] Flexiant Cloud Provider online available at: <http://www.flexiant.com/>

- [45] Li, Ang, et al. "CloudCmp: comparing public cloud providers." *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010.
- [46] BENEDICT, S., 2012. Performance issues and performance analysis tools for HPC cloud applications: a survey. Vienna: Springer.
- [47] Memory speed more important for data-intensive applications such DBMSs or MapReduce
- [48] G. Kousiouris, T. Cucinotta, T. Varvarigou, The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks, *The Journal of Systems and Software*, vol. 84, 2011, pp. 1270-1291.
- [49] https://www.usenix.org/sites/default/files/conference/protected-files/ou_hotcloud12_slides.pdf
- [50] Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. An analysis of performance interference effects in virtual environments. In *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 200–209, April 2007.
- [51] PaaS/IaaS Metamodeling Requirements and SOTA online available at: www.neotextus.net/papers/ispass07/
- [52] Dixit, Kaivalya M. "Overview of the SPEC Benchmarks." (1993): 489-521.
- [53] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun. Benchmarking in the Cloud: What It Should, Can, and Cannot Be. In R. Nambiar and M. Poess, editors, *Selected Topics in Performance*
- [54] Brown, Aaron B., et al. "Benchmarking autonomic capabilities: Promises and pitfalls." *Autonomic Computing, 2004. Proceedings. International Conference on*. IEEE, 2004.
- [55] Evaluation and Benchmarking, volume 7755 of *Lecture Notes in Computer Science*, pages 173–188. Springer Berlin Heidelberg, 2012
- [56] Carsten Binnig, Donald Kossmann, Tim Kraska, and Simon Loesing. How is the Weather tomorrow? Towards a Benchmark for the Cloud. In *Proceedings of the 2nd International Workshop on Testing Database Systems (DBtEST '09)*, Providence, Rhode Island, June 2009.
- [57] Cloud Service Measurement Index Consortium (CSMIC), SMI Framework, <http://www.cloudcommons.com/web/cc/SMIintro>
- [58] Standard Performance Evaluation Corporation online available at: <http://www.spec.org/>
- [59] European Telecommunications Standards online available at: <http://www.etsi.org/>
- [60] J. Gray, *Database and Transaction Processing Performance Handbook*. www.benchmarkresources.com/handbook, 1993.

- [61]YCSB benchmark suite online available at: http://research.yahoo.com/Web_Information_Management/YCSB
- [62]Chohan, Navraj, et al. "Appscale: Scalable and open appengine application development and deployment." *Cloud Computing*. Springer Berlin Heidelberg, 2010. 57-70.
- [63]Open benchmarking online available at: openbenchmarking.org
- [64]Cloudharmony benchmarking tool online available at: cloudharmony.com
- [65]<https://cloudsleuth.net/>
- [66]Iosup, Alexandru, et al. "Performance analysis of cloud computing services for many-tasks scientific computing." *Parallel and Distributed Systems, IEEE Transactions on* 22.6 (2011): 931-945. conferences.sigcomm.org/imc/2010/papers/p1.pdf
- [67]Iosup, Alexandru, Radu Prodan, and Dick Epema. "IaaS cloud benchmarking: approaches, challenges, and experience." *HotTopiCS*. 2013.
- [68]S.K. Garg, S. Versteeg, R. Buyya, "a Framework for
- [69]Ranking of Cloud Computing Services," *Future Generation Computer Systems*, Vol. 29, No. 4, pp. 1012-1023, 2013.
- [70]Chunjie Luo, Jianfeng Zhan, Zhen Jia, Lei Wang, Gang Lu, Lixin Zhang, Cheng-Zhong Xu, Ninghui Sun: CloudRank-D: benchmarking and ranking cloud computing systems for data processing application.
- [71]Bienia, S. Kumar, J. P. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, Oct 2008.
- [72]S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: A benchmarksuite for heterogeneous computing. In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC '09)*, pages 44–54, Austin, TX, USA, 2009. IEEE.
- [73]BENEDICT, S., 2012. Performance issues and performance analysis tools for HPC cloud applications: a survey. Vienna: Springer.
- [74]Nasa Advanced Supercomputing Division available online at: <http://www.nas.nasa.gov/publications/npb.html>
- [75]MathWorks available online at: <http://www.mathworks.com/help/matlab/ref/bench.html>
- [76]The Landscape of Parallel Computing Research: A View from Berkeley available at: <http://eugen.leitl.org/comp/EECS-2006-183.pdf>
- [77]Filebench Home Page - <http://sourceforge.net/apps/mediawiki/filebench/index.php?title=Filebench>

- [78] Filebench WML -
http://sourceforge.net/apps/mediawiki/filebench/index.php?title=Workload_Model_Language ti-
- [79] CloudSuite benchmark suite online available at:
<http://parsa.epfl.ch/cloudsuite/cloudsuite.html>
- [80] DaCappo Benchmarking Suite, Available at:
<http://www.dacapobench.org/>
- [81] Mark Lee Badger, Timothy Grance, Robert Patt-Corner, and Jeffery M. Voas. Cloud Computing Synopsis and Recommendations. Technical report, NIST Computer Security Division, 2012
- [82] Fuentes, L.; Vallecillo, A. 2004. An introduction to UML profiles. UPGRADE, The European Journal for the Informatics Professional, v.5, n.2, pp. 6-13.
- [83] Poole, John D. "Model-driven architecture: Vision, standards and emerging technologies." *Workshop on Metamodeling and Adaptive Object Models, ECOOP*. Vol. 2001. 2001.
- [84] UML and Model Engineering online available at:
<http://www.cepis.org/files/cepisupgrade/full-2004-II.pdf>
- [85] Kleppe, Anneke G., Jos B. Warmer, and Wim Bast. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.
- [86] Object Management Group (OMG) online available at:
<http://www.omg.org/>
- [87] Unified Modeling Language (UML) <http://www.uml.org/#UMLProfiles>
- [88] Object Management Group. Meta Object Facility (MOF) Specification. OMG document: formal/2002-04-03. 2003.
- [89] Bran Selic. MDA Manifestations. UPGRADE: The European Journal for the Informatics Professional, 9(2):12–16, 2008.
- [90] Manuel Wimmer, Gertti Kappel, Angelika Kusel, Werner Retschitzegger, Johannes Schönböck, and Wieland Schwinger. Fact or Fiction – Reuse in Rule-Based Model-to-Model Transformation Languages. In Proc. Intl. Conf. on Theory and Practice of Model Transformations (ICMT), pages 280–295, 2012.
- [91] R. Sharma and M. Sood, "Enhancing Cloud SaaS Development With Model Driven Architecture", *International Journal on Cloud Computing: Services and Architecture*, 2011.
- [92] Sharma, Ritu, and Manu Sood. "Cloud SaaS and model driven architecture." *International Conference on Advanced Computing and Communication Technologies (ACCT11)*. 2011.
- [93] Ricardo Pérez-Castillo, Ignacio García Rodríguez de Guzmán, and Mario Piattini. Knowledge Discovery Metamodel-ISO/IEC 19506: A Standard to

- Modernize Legacy Systems. *Computer Standards & Interfaces*, 33(6):519–532, 2011.
- [94] REMICS project-available online at:
http://www.remics.eu/system/files/REMICS_D6.6.lowres.pdf
- [95] Eirik Brandzæg, Parastoo Mohagheghi, and Sébastien Mosser. Towards a Domain-Specific Language to Deploy Applications in the Cloud. In *Proc. Intl. Conf. on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING)*, pages 213–218, 2012.
- [96] Glauco Gonçalves, Patricia Endo, Marcelos Santos, Djamel Sadok, Judith Kelner, Bob Merlander, and Jan-Erik Mångs. CloudML: An Integrated Language for Resource, Service and Request Description for D-Clouds. In *Proc. Intl. Conf. on Cloud Computing Technologies and Science (CloudCom)*, pages 399–406, 2011.
- [97] <http://www.omg.org/spec/SysML/1.3/>
- [98] Sanford Friedenthal, Alan Moore, Rick Steiner: *OMG Systems Modeling Language (OMG SysML) Tutorial*, 2008.
- [99] Sanford Friedenthal, Alan Moore, Rick Steiner: *OMG Systems Modeling Language (OMG SysML) Tutorial*, 2008.
- [100] <http://www.w3.org/Math/>
- [101] Object Constraint Language available at:
<http://www.omg.org/spec/OCL/2.0/>
- [102] FUML language description available at:
<http://www.omg.org/spec/FUML/1.0/>, 2011
- [103] B. Selic. The less well known UML. In *Formal Methods for MDE*, volume 7320 of LNCS, pages 1-20. Springer Berlin / Heidelberg, 2012.
- [104] <http://www.modelexecution.org>
- [105] Mayerhofer, T., Langer, P., and Wimmer, M. Towards xMOF: Executable DSMLs based on fUML. In *Proceedings of the 12th Workshop on Domain-Specific Modeling (DSM'12) at SPLASH 2012*.
- [106] Filippo Bosi et al, Cloud4SOA Semantic Layer, Cloud4SOA deliverable, 2011.
- [107] O. Corcho, M. Fernández-lópez, A. Gómez-pérez, and A. López, "Building legal ontologies with METHONTOLOGY and WebODE," in *Law and the Semantic Web*, number 3369 in LNAI: Springer-Verlag, 2005, pp. 142--157.
- [108] The Open Group - SOA WG Open SOA Ontology TC – “Service-Oriented Architecture Ontology” – TS C104, October 2010
- [109] Essential Meta-Model – an ontology for the domain of enterprise architecture – <http://www.enterprise-architecture.org/about/35-essential-meta-model>

- [110] Togaf 9 Core Content Metamodel – An OWL Ontology for the TOGAF 9 Core ContentMetamodel - <http://sites.google.com/site/ontologyprojects/home/togaf-core-content-metamodel>
- [111] Nguyen, D.K., Lelli, F., Taher, Y., Parkin, M., Papazoglou, M.P., van den Heuvel, W.-J.: Blueprint Template Support for Engineering Cloud-Based Services. In: Abramowicz, W., Llorente, I.M., SurrIDGE, M., Zisman, A., Vayssière, J. (eds.) ServiceWave 2011. LNCS, vol. 6994, pp. 26–37. Springer, Heidelberg (2011).
- [112] Hashemi, Seyyed Mohsen, and Amid Khatibi Bardsiri. "Cloud computing Vs. grid computing." ARPN J. Syst. Softw 2.5 (2012): 188-194
- [113] Virtualization definition available at: <http://yoyoclouds.wordpress.com/2012/04/24/what-is-virtualization/>
- [114] Intel Cloud Computing Taxonomy and Ecosystem Analysis available at: <http://www.intel.com/content/dam/doc/case-study/intel-it-cloud-computing-taxonomy-ecosystem-analysis-study.pdf>
- [115] Djurić, Dragan, Dragan Gašević, and Vladan Devedžić. "The Tao of Modeling Spaces." JOURNAL OF OBJECT TECHNOLOGY 5.8.
- [116]] Ritu Sharma and Manu Sood: Cloud SaaS: Models and Transformation. In: Advances in Digital Image Processing and Information Technology. Communications in Computer and Information Science, 2011, Volume 205, Part 2, 305-314, DOI: 10.1007/978-3-642-24055-3_31
- [117] The Fast Guide to Model Driven Architecture(OMG) online available at: http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf
- [118] SOTA in Modeling languages and Model Transformation Techniques online available at: http://www.artist-project.eu/sites/default/files/D9.1%20SOTA%20in%20modeling%20languages%20and%20model%20transformationtechniques_M6_3103_2013.pdf
- [119] Cloud Services and Performance Analysis Framework online available at: http://www.artist-project.eu/sites/default/files/D7.2.1%20Cloud%20services%20modeling%20and%20performance%20analysis%20framework_M12_30092_013.pdf