Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

The Cloud Computing Paradigm: Characteristics, Opportunities and Research Issues

R. Giordanelli, C. Mastroianni

RT-ICAR-CS-10-01 **Aprile 2010**

Consiglio Nazionale delle Ricerche – Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sedi di Cosenza (*Via P. Bucci 41C, 87036 Rende, Italy*), di Napoli (*Via P. Castellino 111, 80131 Napoli*) e Palermo (*Viale delle Scienze, Parco D'Orleans, 90128 Palermo*)– URL: *www.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

The Cloud Computing Paradigm: Characteristics, Opportunities and Research Issues

R. Giordanelli, C. Mastroianni

Rapporto Tecnico N.: **RT-ICAR-CS-10-01**

Data: **Aprile 2010**

The Cloud Computing Paradigm: Characteristics, Opportunities and Research Issues

Raffaele Giordanelli, Carlo Mastroianni
ICAR-CNR, Rende(CS), Italy
{giordanelli, mastroianni}@icar.cnr.it

Abstract

The Cloud Computing paradigm has become one of the most interesting trends in the arena of distributed computing owing to its numerous merits: Cloud frameworks allow the managers of server farms to adapt and optimize resource usage; the "start-up" of small and medium enterprises is becoming easier and less onerous, thanks to the possibility of cutting fixed costs; Clouds can support industrial and academic research efficiently, because they allow complex tasks to be executed in short time with the use of resources allocated on demand. After a short summary of Cloud Computing origins and history, this document aims to illustrate the main ideas underlying the Cloud paradigm, and describe the different kinds of Cloud frameworks, trying to shed light over the confusion about the Cloud Computing concept. This document also examines the peculiarities and similarities of Cloud Computing with respect to the other paradigms that have recently emerged in the field of parallel/distributed computing, such as Peer-to-Peer, Grid Computing and Utility Computing. All these paradigms have several common objectives, therefore there are notable benefits that can derive from the use of Cloud solutions in existing and future computing systems: these benefits, and the way to accomplish them, are discussed. Finally, the most important Cloud frameworks that are currently available are illustrated and compared.

Contents

1 What is Cloud Computing

Cloud Computing has the potential to transform a large part of the IT industry, shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require large capital outlays to deploy their facilities and provide their services. They do not need to be concerned about over-provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under-provisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1000 servers for one hour costs no more than using one server for 1000 hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT [3].

Cloud Computing is a paradigm hinting at a future in which we will not compute only on local computers, nor on clusters maintained by the same company or organization, but on computing facilities and storage utilities operated by third-parties [13].

The term *cloud* has long been used as a metaphor for the Internet and local subnets, based on how these are depicted in computer network diagrams, and it is used as an abstraction of the underlying infrastructure it conceals. In fact, the concept of Cloud Computing is strictly connected to the concepts of Web and Web services, of which it can be considered the most recent extension. Typical Cloud Computing services provide online business applications that can be accessed from a Web browser, while the software and data are stored on the servers.

This vision brings concepts of economies of scale into the problem of computing, since it allows small and mid-sized companies to outsource their data-center infrastructures, and larger companies to get peak load capacity without the necessity of building larger data centers internally [16].

The value of the ideas behind the concept of Cloud Computing is testified by the presence of all the giants in the IT field. We can see Microsoft, IBM, Google, Amazon, Sun Microsystems, Nokia, striving to get a dominant position, after an independent company (Salesforce) opened the path towards the Cloud Computing solutions. This document will analyze some of the proposed technologies.

According to Ian Foster, Cloud Computing is:

> *A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet [13].*

There are a few key points in this definition. First, Cloud Computing is a specialized distributed computing paradigm; it differs from traditional ones in that 1) it is massively scalable, 2) it can be encapsulated as an abstract entity that delivers different levels of services to customers outside the Cloud, 3) it is driven by economies of scale, and 4) the services can be dynamically configured (via virtualization or other similar approaches) and delivered on demand.

There are three main factors contributing to the surge of interest in Cloud Computing: 1) rapid decrease in hardware cost and increase in computing power and storage capacity, and the advent of multi-core architectures and modern supercomputers consisting of hundreds of thousands of cores; 2) the exponentially growing data size in scientific instrumentation/simulation and Internet publishing and archiving, and 3) the wide-spread adoption of Services Computing and Web 2.0 applications.

After this brief introduction, Section 2 will give a short account on Cloud Computing history. Section 3 will give an overview about the different classes of Cloud Computing systems, introducing concepts like Infrastructure as a System (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Section 4 will show the main features of Cloud Computing systems, in terms of business model, architecture,

resource management, programming model, and security model. Section 5 will illustrate the differences, similarities and possible integration between Cloud Computing and other distributed paradigms, such as Peer-to-Peer (P2P), Grid and Cluster Computing. Section 6 will explain the advantages in using Cloud Computing Systems. Section 7 will describe the current software solutions offered by cloud vendors. Section 8 will draw some conclusions about the current state and the future of Cloud Computing.

2 History

The *Cloud* term is borrowed from telephony. Up to the 1990s, data circuits (including those carrying Internet traffic) were hard-wired. Then, long-haul telephone companies began offering Virtual Private Network (VPN) services for data communications, with guaranteed bandwidth and at a lower cost because they could switch traffic to balance utilization, thus utilizing network bandwidth more efficiently. As a result of this arrangement, it was impossible to determine in advance which paths the traffic would be routed over. The term "telecom cloud" was used to describe this type of networking, and Cloud Computing is a somewhat similar concept .

The concept of Cloud Computing dates back to 1961, when John McCarthy opined that "computation may someday be organized as a public utility" [19]. In 1997, the first academic definition was provided by Ramnath K. Chellappa, who called it a computing paradigm where the boundaries of computing will be determined by economic rationale rather than technical limits [5]. The term cloud had already come into commercial use in the early 1990s to refer to large Asynchronous Transfer Mode (ATM) networks. By the turn of the 21st century, the term "Cloud Computing" began to appear more widely [18], although most of the focus at that time was limited to SaaS, or *Software as a Service*.

In 1999, Salesforce.com was established by Marc Benioff, Parker Harris, and their associates. They applied many technologies developed by companies such as Google and Yahoo! to business applications. They also provided "on demand" computing facilities, or SaaS, to their customers.

In the early 2000s, Microsoft extended the concept of SaaS through the development of Web services. IBM detailed these concepts in 2001 in the Autonomic Computing Manifesto [15], which described advanced automation techniques such as self-monitoring, self-healing, self-configuring, and self-optimizing in the management of complex IT systems with heterogeneous storage, servers, applications, networks, security mechanisms, and other system elements that can be virtualized across an enterprise. In 2007, Google, IBM, and a number of universities embarked on a large scale Cloud Computing research project [17].

These days, Cloud Computing has become a popular buzzword: Figure 1 shows the popularity of this term compared to "Utility Computing", "Distributed Computing" and "Grid Computing" terms. A side effect of this hype is that it is often difficult to distinguish what is Cloud Computing from what is not.

Not all people are enthusiastic about Cloud, here two significant quotes are reported:

> *I don't understand what we would do differently in the light of Cloud Computing other than change the wording of some of our ads (Larry Ellison, Oracles CEO)*
>
> *Somebody is saying this is inevitable and whenever you hear somebody saying that, it's very likely to be a set of businesses campaigning to make it true (Richard Stallman)*

Among all possible quotes, these ones pose the focus on the major obstacles to the massive use of Cloud Computing: the difficult interaction between Cloud Computing and (relational) data, and the privacy of data in the Clouds.

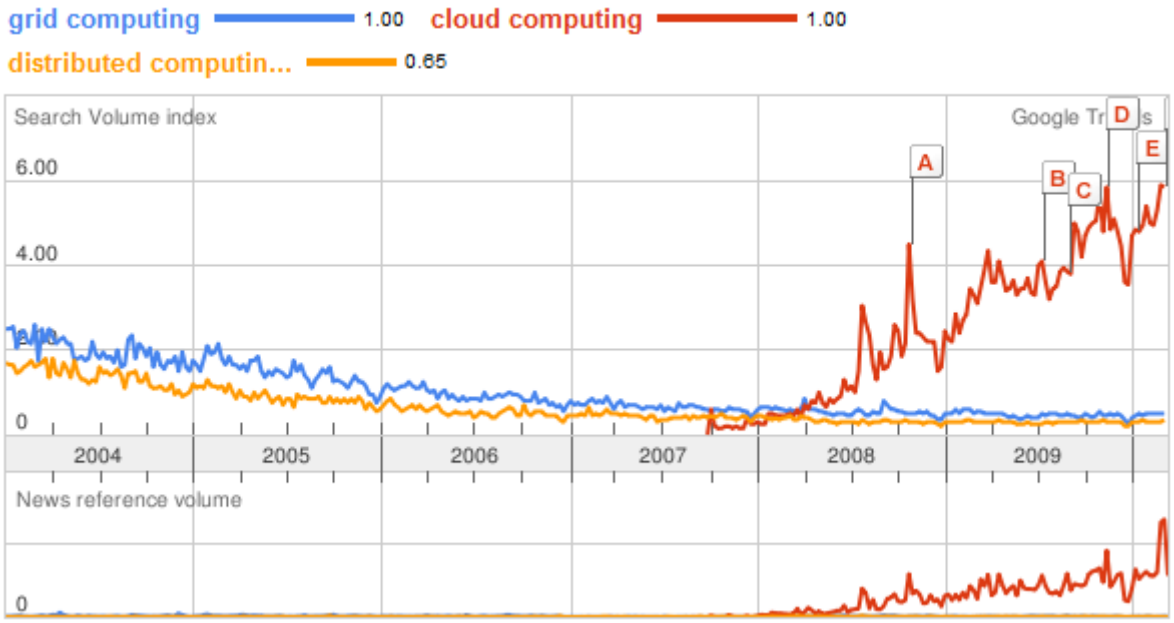

Figure 1. Popularity of Cloud Computing, as reported by Google trends.

3 Different kinds of Cloud Computing

There is a widespread confusion about what Cloud Computing really is [14]. The first difficulty is to understand that Cloud Computing contains 3 different paradigms in itself [22, 20, 13].

SaaS, Software as a Service , provides a complete, turnkey application that many users sign-up for and use without any concern about where, how, by whom the compute cycles and storage bits are provided. The service being sold is a complete end-user application. This is what almost everyone has already used in the form of gmail, yahoo mail, wordpress.com, Quicken Online, the rest of google apps, the various search engines, social networks, wikipedia, encyclopedia britannica, and is the major source of confusion about what Cloud Computing is, because every service published in the Internet falls into this category.

PaaS, Platform as a Service , provides a full or partial application development environment that users can access and utilize online, even in collaboration with others. This is the newest entry where an application platform is offered to developers in the Cloud. Developers write their application to a more or less open specification and then upload their code into the cloud where the application is run, typically being able to scale up automatically as usage for the application grows. Examples are Mosso, Heroku, Google App Engine, Engine Yard, Joyent and Force.com (SalesForce platform). The service being sold is the machinery that funnels requests to an application and makes the application tick.

IaaS, Infrastructure as a Service ,,, (also HaaS, Hardware as a Service) provides a full computer infrastructure via the Internet (Amazon's EC2, Eucalyptus, GoGrid, RightScale and Linode). This is the most general offering that Amazon has pioneered and where RightScale offers its management platform. Developers and system administrators obtain general compute, storage, queueing, and other resources and run their applications with the fewest limitations. This is the most powerful

type of Cloud in that virtually any application and any configuration fits the Internet can be mapped to this type of service. Of course it also requires more work on the part of the buyer, which is where RightScale comes in to help with set-up and automation.

Looking at these different types of Clouds it is pretty clear that they are geared toward different purposes and that they all have a reason for being. The platforms in the Cloud are a very interesting offering in that they promise to take some of the pain away from dealing with the raw infrastructure.

3.1 Cloud Pyramid

Michael Sheehan proposed a *Cloud Pyramid* (Figure 2) to help differentiate the various Cloud offerings [20]:

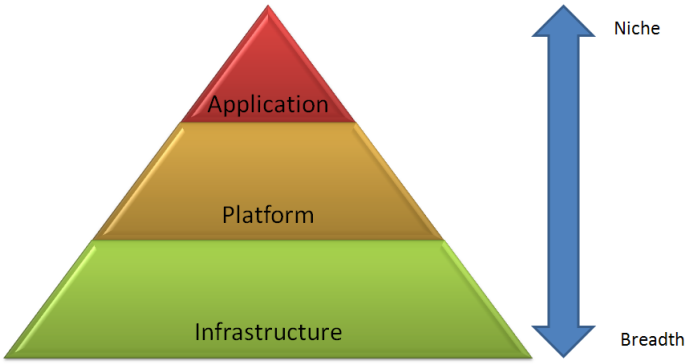

Figure 2. The Cloud Pyramid. (from "Introducing the Cloud Pyramid" [20])

Cloud Application Within this part of the pyramid, users are truly restricted to only what the application is and can do (SaaS). Its main characteristics are easiness to use and access and good consumer adoption. The weaknesses are the impossibility of customizing the services, as the application can be used as far as what it is designed for; no control or knowledge of underlying technology.

Cloud Platforms As you move further down the pyramid, the user gains increased flexibility and control but is still fairly restricted (PaaS). This category is becoming more congested with competitors, many of whom are trying to leverage the Cloud Infrastructure. Its main characteristic is that it is *developers oriented*: it is possible to upload a tightly configured application and make it run. The developers have more control than with a Cloud Application. The main weakness is that who uses a Cloud Platform is restricted to the platform's ability only, and it is hard to work "outside the box".

Cloud Infrastructure At the bottom of the pyramid are the infrastructure providers (IaaS) like Amazon's EC2, GoGrid, RightScale and Linode. Companies providing a Cloud Infrastructure may enable the deployment od Cloud Platforms and Cloud Applications. Most companies within this segment operate their own infrastructure, allowing them to provide more features, services and control within the pyramid. The main characteristic is the full control of server infrastructure, not confined to "containers", "applications" or restrictive instances.

There are other ways to display this hierarchy, for example, if one were to weight the graphic by the number of providers within each segment, the pyramid would be upside-down. The point here though is to show how these Cloud segments build upon and are somewhat dependent upon each other. While they are directly related, they do not require interdependence (e.g., a Cloud Application does not necessarily have to be built upon a Cloud Platform or a Cloud Infrastructure).

4 Cloud Features

This section discusses the main features of Cloud Computing systems. It starts by illustrating the Cloud Computing business model, which is different from that of usual computing systems. Then the software architecture of Clouds is shown and the main aspects of Cloud Computing systems are explained, such as resource management, programming model and security.

4.1 Business Model

The traditional business model for software has been a one-time payment for unlimited use (usually on a single computer) of the software. In a Cloud-based business model, a customer will pay the provider on a consumption basis, very much like the utility companies charge for basic utilities such as electricity, gas, and water, and the model relies on economies of scale in order to drive prices down for users and profits up for providers. Today, Amazon essentially provides a centralized Cloud consisting of Compute Cloud EC2 and Data Cloud S3. The former is charged based on per instance-hour consumed for each instance type and the later is charged by per GB-Month of storage used. In addition, data transfer is charged by TB/month data transfer, depending on the source and target of such transfer.

The business model for Grids (at least the one found in academia or government labs) is project-oriented: users are provided a given number of service units (i.e. CPU hours) to spend. What makes an institution wish to join the TeraGrid? When an institution joins the TeraGrid offering a set of resources, it knows that others in the community can now use these resources across the country. It also acknowledges the fact that it gains access to a dozen other Grid sites.

4.2 Architecture

Grids started off in the mid-90s to address large-scale computation problems using a network of resource-sharing commodity machines that deliver the computation power affordable only by supercomputers and large dedicated clusters at that time.

Clouds are developed to address Internet-scale computing problems where some assumptions are different from those of the Grids. Clouds are usually referred to as a large pool of computing and/or storage resources, which can be accessed via standard protocols via an abstract interface. Clouds can be built on top of many existing protocols such as Web Services (WSDL, SOAP), and some advanced Web 2.0 technologies such as REST, RSS, AJAX, etc. In fact, behind the cover, it is possible for Clouds to be implemented over existing Grid technologies leveraging more than a decade of community efforts in standardization, security, resource management, and virtualization support.

There are multiple definitions of the Cloud architecture. Among them we consider a four-layer architecture for Cloud Computing, similar to that used for the Grid architecture, composed of 1) fabric, 2) unified resource, 3) platform, and 4) application layers.

The fabric layer contains the raw hardware level resources, such as compute resources, storage resources, and network resources. The unified resource layer contains resources that have been abstracted/encapsulated (usually by virtualization) so that they can be exposed to upper layer and end

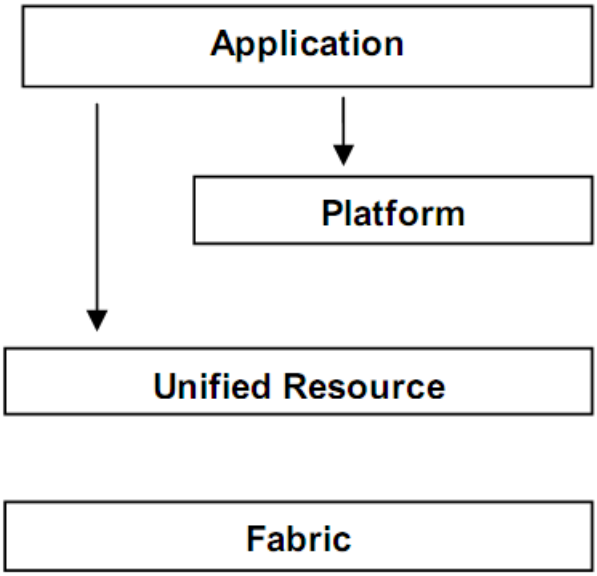

Figure 3. Cloud architecture. (from "What is the grid? a three point checklist" [12].)

users as integrated resources, for instance, a virtual computer/cluster, a logical file system, a database system, etc. The platform layer adds on a collection of specialized tools, middleware and services on top of the unified resources to provide a development and/or deployment platform. For instance, a Web hosting environment, a scheduling service, etc. Finally, the application layer contains the applications that run in the Clouds.

Although Clouds provide services at three different levels (IaaS, PaaS, and Saas), standards for interfaces to these different levels still remain to be defined. This leads to interoperability problems between Clouds, and there is little business incentives for Cloud providers to invest additional resources in defining and implementing new interfaces. As Clouds mature, and more sophisticated applications and services emerge that require the use of multiple Clouds, there will be growing incentives to adopt standard interfaces that facilitate interoperability in order to capture emerging and growing markets in a saturated Cloud market.

4.3 Resource Management

This section describes resource management in Grid and Clouds, covering topics such as the computing model, data model, virtualization, and security. These topics are extremely important to understand the main challenges that Clouds face today, and will have to be overcome in the future.

4.3.1 Compute Model

Most Grids use a batch-scheduled compute model, in which a local resource manager (LRM), such as Condor, manages the compute resources for a Grid site, and users submit batch jobs (via GRAM) to request some resources for a given time interval.

The Cloud computing model is different, because resources in the Cloud are shared by all users at the same time (in contrast to dedicated resources governed by a queuing system). This should allow latency sensitive applications to operate natively on Clouds, although ensuring a good level of QoS for end users

will not be trivial, and will likely be one of the major challenges for Cloud Computing as the Clouds grow in scale and number of users.

4.3.2 Data Model

Cloud Computing and local computing will coexist and evolve hand in hand, while data management (mapping, partitioning, querying, movement, caching, replication, etc) will become more and more important for both Cloud Computing and local computing with the increase of data-intensive applications. The critical role of Cloud Computing goes without saying, but the importance of local computing cannot be overlooked for several reasons:

- For security reasons, people might not be willing to run mission-critical applications on the Cloud and send sensitive data to the Cloud for processing and storage.
- Users want to get their things done even when the Internet and Cloud are down or the network communication is slow.
- With the advances of multi-core technology, the coming decade will bring the possibilities of having a desktop supercomputer with hundreds to thousands of threads/cores.

The importance of data has caught the attention of the Grid community for the past decade; Data Grids have been specifically designed to tackle data intensive applications in Grid environments.

A huge obstacle to the massive use of Cloud Computing is the data transfer bottleneck, and the related costs. At $100 to $150 per terabyte transferred, these costs can quickly add up, making data transfer costs an important issue [3]. One opportunity to overcome the high cost of Internet transfers is to ship disks. To quantify the argument, assume that we want to ship 10 TB from U.C. Berkeley to Amazon in Seattle, Washington. Using Amazon S3, it would take more than 45 days. If we instead sent ten 1TB disks via overnight shipping, it would take less than one day to transfer 10TB [3].

4.3.3 Data Locality

As CPU cycles become cheaper and data sets double in size every year, the main challenge for efficient scaling of applications is the location of the data related to the available computational resources: moving the data repeatedly to distant CPUs is becoming the bottleneck. To achieve good scalability at Internet scales for Clouds, Grids, and their applications, data must be distributed over many computers, and computations must be steered towards the best place to execute in order to minimize the communication costs. An example is Google's MapReduce [8] system, which runs on top of the Google File System, within which data is loaded, partitioned into chunks, and each chunk replicated. Thus data processing is collocated with data storage: when a file needs to be processed, the job scheduler consults a storage metadata service to get the host node for each chunk, and then schedules a map process on that node, so that data locality is exploited efficiently. Even more critical is the combination of the compute and data resource management, which leverages data locality in access patterns to minimize the amount of data movement and improve end-application performance and scalability. It is important to schedule computational tasks close to the data, and to understand the costs of moving the work as opposed to moving the data. Data-aware schedulers and dispersing data close to processors is critical in achieving good scalability and performance.

4.3.4 Virtualization

Virtualization has become an indispensable ingredient for Clouds, to support abstraction and encapsulation. Just like threads were introduced to provide users the illusion that the computer were running all the threads simultaneously, and each thread were using all the available resources, Clouds need to run multiple (or even up to thousands or millions of) user applications, and all the applications appear to the users as if they were running simultaneously and they could use all the available resources in the Cloud.

Virtualization provides the necessary abstraction such that the underlying fabric (raw compute, storage, network resources) can be unified as a pool of resources and resource overlays (e.g. data storage services, Web hosting environments) can be built on top of them. Virtualization also enables each application to be encapsulated such that they can be configured, deployed, started, migrated, suspended, resumed, stopped, etc., and thus provides better security, manageability, and isolation.

There are also many other reasons why Clouds tend to adopt virtualization: 1) server and application consolidation: as multiple applications can be run on the same server, resources can be utilized more efficiently; 2) configurability: requirements for various applications could differ significantly, for example storage and computing resources. As it is not possible to dynamically configure and bundle (aggregate) resources at the hardware level, virtualization is necessary; 3) increased application availability: virtualization allows quick recovery from unplanned outages, as virtual environments can be backed up and migrated with no interruption in service; 4) improved responsiveness: resource provisioning, monitoring and maintenance can be automated, and common resources can be cached and reused.

All these features of virtualization provide the basis for Clouds to meet stringent SLA (Service Level Agreement) requirements in a business setting. This cannot be achieved easily in a non-virtualized environment, as systems would have to be over-provisioned to handle peak load and would waste resources in idle periods. After all, a virtualization infrastructure can be just thought as a mapping from IT resources to business needs.

It is also worth noting that in the past virtualization leaded to significant performance losses for applications. However, over the past few years, processor manufacturers such as AMD and Intel have been introducing hardware support for virtualization, which is helping to reduce the gap between performance of applications executed on virtualized resources and on traditional operating systems.

Another challenge brought by virtualization is the potential difficulty in fine-control over the monitoring of resources. In a Cloud, different levels of services can be offered to an end user: the user is only given access to a pre-defined API, and the lower level resources are opaque to the user (especially at the PaaS and SaaS levels). The user does not have the freedom to deploy her own monitoring infrastructure, and the limited information returned to her may not provide the necessary level of detail to figure out which is the state of a resource. On the other hand, monitoring can be argued to be less important in Clouds, as users are interacting with a more abstract layer that is potentially more sophisticated; this abstract layer could respond to failures and quality of service (QoS) requirements automatically and in a general-purpose way, irrespective of application logic.

4.4 Programming Model

Programming in a Cloud is different from classical programming. The main programming technologies used in Clouds are described in the following.

MapReduce [8] is a parallel programming model tailored to the processing of large datasets, and it is based on a simple model with just two key functions: *map* and *reduce*, borrowed from functional languages. The *map* function applies a specific operation to each of a set of items, producing a new set of items; the *reduce* function performs aggregation on a set of items. The MapReduce runtime

system automatically partitions input data and schedules the execution of programs in a large cluster of commodity machines. The system is made fault tolerant by checking worker nodes periodically and reassigning failed jobs to other worker nodes. Hadoop [2] is the open source implementation of the MapReduce model.

Mesh-ups and scripting (Java Script, PHP, Python etc) are taking the place of a workflow system in the Cloud world, since there is no easy way to integrate services and applications from various providers. They are essentially data integration approaches, because they take outputs from one service/application, transform them and feed into another.

Clouds have generally adopted Web Services APIs where users access, configure and program Cloud applications using pre-defined APIs exposed as Web services, and HTTP and SOAP are the common protocols chosen for such services. Although Clouds adopted some common communication protocols such as HTTP and SOAP, the integration and interoperability of all the services and applications remain the biggest challenge, as users need to tap into a federation of Clouds instead of a single Cloud provider.

4.5 Security Model

Clouds mostly comprise dedicated data centers belonging to the same organization. Within each data center, hardware and software configurations and supporting platforms are in general more homogeneous when compared to those in Grid environments. Interoperability can become a serious issue for cross-data center, cross-administration domain interactions: imagine running your accounting service in Amazon EC2 while other business operations are on the Google infrastructure. Currently, the security model for Clouds seems to be relatively simpler and with lower guarantees than the security model adopted by Grids. Cloud infrastructure typically relies on Web forms (over SSL) to create and manage account information for end-users, and allows users to reset their passwords and receive new passwords via email in an unsafe and unencrypted communication. Note that new users can use Clouds relatively easily and almost instantly, with a credit card and/or email address. Security is one of the largest concerns for the adoption of Cloud Computing. Seven risks that a Cloud user should take into account before committing with vendors have been outlined [13, 4]:

- Privileged user access: sensitive data processed outside the enterprise needs the assurance that they are only accessible and propagated to privileged users;
- Regulatory compliance: a customer needs to verify if a Cloud provider has external audits and security certifications and if the infrastructure complies with some regulatory security requirements;
- Data location: since a customer will not know where her data will be stored, it is important that the Cloud provider commits to storing and processing data in specific jurisdictions and to obey local privacy requirements on behalf of the customer;
- Data segregation: the customer's data must be fully segregated from other customers' data;
- Data recovery: it is important that the Cloud provider has an efficient replication and recovery mechanism to restore data if a disaster occurs;
- Investigative support: Cloud services are especially difficult to investigate. If this is important for a customer, then such support needs to be ensured with a contractual commitment;
- Long-term viability: your data should be viable even if the Cloud provider is acquired by another company.

5 Cloud Computing and other paradigms

This section will discuss other paradigms that are similar or have some relationship with Cloud Computing. Some of them share the same visions and purposes, others have been useful to the realization of Clouds.

5.1 Cloud Computing and Grid Computing

Cloud Computing partially overlaps with Grid Computing: in some sense it evolved out of Grid Computing and uses Grid Computing as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources (such is the case in Grids) to one that is economy-based aiming to deliver more abstract resources and services (such is the case in Clouds).

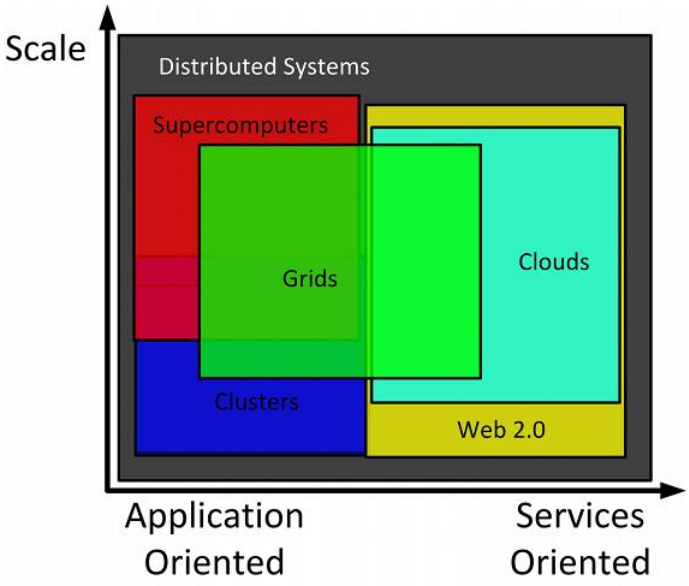

Figure 4. Grids and Clouds overview. (from "What is the grid? a three point checklist" [12].)

Ian Foster gave a three point checklist [12] to help define what is, and what is not a Grid: 1) the Grid coordinates resources that are not subject to centralized control, 2) the Grid uses standard, open, general-purpose protocols and interfaces, and 3) the Grid delivers non-trivial qualities of service. Although point 3 holds true for Cloud Computing, neither point 1 nor point 2 is clear that it is the case for today's Clouds.

5.2 Clusters

A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks. Clusters are usually deployed to improve performance and/or availability with respect to a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Clusters can be categorized in three classes:

High-availability (HA) clusters High-availability clusters (also known as Failover Clusters) are implemented primarily for the purpose of improving the availability of services that the clusters provide. They operate by having redundant nodes, which are then used to provide services when system components fail. The most common size for an HA cluster is two nodes, which is the minimum requirement to provide redundancy. HA cluster implementations attempt to use redundancy of cluster components to eliminate single points of failure. There are commercial implementations of High-Availability clusters for many operating systems. The Linux-HA project is one commonly used free software HA package for the Linux operating system.

Load-balancing clusters Multiple computers are linked together to share the computational workload, but are seen from the user side as a single virtual computer. User requests are managed by, and distributed among, all the stand-alone computers to form a cluster. This results in balanced computational work among different machines, improving the performance of the cluster system.

Compute clusters Often clusters are used primarily for computational purposes, rather than handling IO-oriented operations such as Web services or databases. For instance, a cluster might support computational simulations of weather or vehicle crashes. The primary distinction within compute clusters is how tightly-coupled the individual nodes are. For instance, a single compute job may require frequent communication among nodes - this implies that the cluster shares a dedicated network, is densely located, and probably has homogenous nodes. This cluster design is usually referred to as Beowulf Cluster. The other extreme is where a compute job uses one or few nodes, and needs little or no inter-node communication. Tightly-coupled compute clusters are designed for work that might traditionally have been called "supercomputing". Middleware such as MPI (Message Passing Interface) or PVM (Parallel Virtual Machine) permits compute clustering programs to be portable to a wide variety of clusters.

5.3 Peer to Peer

There are several peer-to-peer (P2P) systems tailored to the problem of data storage and distribution. The first generation of P2P systems, such as Freenet and Gnutella, were predominantly used as file sharing systems. These were examples of *unstructured* P2P networks where the overlay links between peers were established arbitrarily. In these networks, a search query is usually flooded through the network to find as many peers as possible that share the data. P2P systems evolved to the next generation into what are widely known as *structured* P2P networks. These networks employ a globally consistent protocol to ensure that any node can efficiently route a search query to a peer that has the desired data. Systems like CAN, Pastry and Chord use routing mechanisms to ensure that queries can be answered within a bounded number of hops, typically logarithmic with the number of peers.

5.3.1 P2P for Clouds: the Amazon Dynamo example

Dynamo is the fundamental distributed storage system for Amazon Web Services, since the other storage services are based on it. Like a relational database it stores information to be retrieved, but it does not break the data into tables. Instead, all objects are stored and looked up via their key. [9]

Dynamo stores objects associated with a key through a simple interface; it exposes two operations: *get* and *put*. The *get(k)* operation locates the object replicas associated with the key k in the storage system, and returns an object or a list of objects along with their context. The *put(k, obj)* operation determines where the replicas of the object *obj* should be placed based on the associated key k, and writes the replicas to disk. Dynamo manages both the key and the object supplied by the caller as an

opaque array of bytes. It applies a MD5 hash to the key to generate a 128-bit identifier, which is used to determine the storage nodes that are responsible for serving the key.

One of the main design requirements for Dynamo is that it must scale incrementally. This requires a mechanism to dynamically partition the data over the set of nodes (i.e., storage hosts) in the system. The Dynamo partitioning scheme relies on consistent hashing to distribute the load across multiple storage hosts. In consistent hashing, the output range of a hash function is managed as a fixed circular space or "ring" (i.e. the largest hash value wraps around to the smallest hash value), as in Chord [21]. Each node in the system is assigned a random value within this space that represents its "position" on the ring. Each data item identified by a key is assigned to a node by hashing the key to yield its position on the ring, and then walking the ring clockwise to find the first node with a position larger than the item's position.

Thus, each node becomes responsible for the region in the ring between it and its predecessor node on the ring. The main advantage of consistent hashing is that the departure or arrival of a node only affects its immediate neighbors, whereas other nodes are not affected. The basic consistent hashing algorithm presents some challenges. First, the random position assignment of each node on the ring leads to non-uniform data and load distribution. Second, the basic algorithm is oblivious to the heterogeneity in the performance of nodes. To address these issues, Dynamo uses a variant of consistent hashing [21]): instead of mapping a node to a single point in the circle, each node is assigned to multiple points in the ring. To this end, Dynamo uses the concept of "virtual nodes". A virtual node looks like a single node in the system, but each node can be responsible for more than one virtual node. Therefore, when a new node is added to the system, it is assigned multiple positions (henceforth, "tokens") in the ring.

This type of P2P systems can be improved with self-organizing features, as shown in Self-Chord [11]. Self-Chord inherits the ability of Chord-like structured systems for the construction and maintenance of an overlay of peers, but features enhanced functionalities deriving from the activity of ant-inspired mobile agents, such as autonomy behavior, self-organization and capacity to adapt to a changing environment.

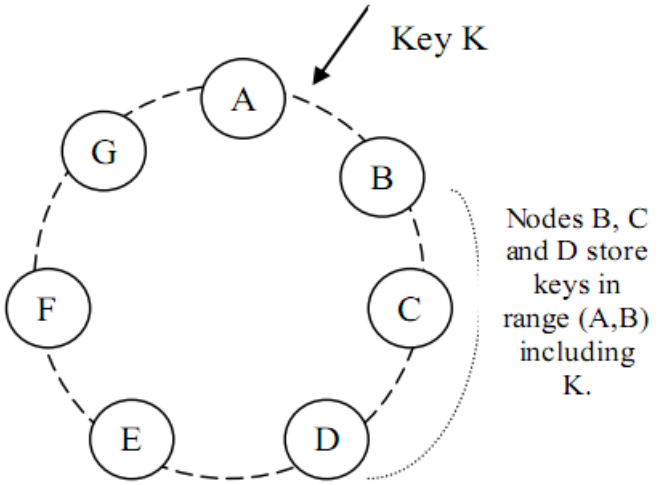

Figure 5. Partitioning and replication of keys in Dynamo ring. (from "Dynamo: Amazon's Highly Available Key-value Store" [9].)

To achieve high availability and durability, Dynamo replicates its data on multiple hosts. Each data

item is replicated at N hosts, where N is a parameter configured "per-instance". Each key k is assigned to a coordinator node. The coordinator is in charge of the replication of the data items that fall within its range. In addition to locally storing each key within its range, the coordinator replicates these keys at the *N-1* clockwise successor nodes in the ring. This results in a system where each node is responsible for the region of the ring between it and its *Nth* predecessor. In Figure 5, node B replicates the key k at nodes C and D in addition to storing it locally. Node D will store the keys that fall in the ranges (A, B], (B, C], and (C, D]. The list of nodes that are responsible for storing a particular key is called the *preference list*. The system is designed so that every node in the system can determine which nodes should be in this list for any particular key.

6 Advantages in Adopting the Cloud

Three aspects pertaining to the hardware configuration are new in Cloud Computing [3].

- The illusion of infinite computing resources available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning.
- The elimination of an up-front commitment by Cloud users, thereby allowing companies to start small, and increase hardware resources only when there is an increase in their needs.
- The ability to pay for the use of computing resources on a short-term basis as needed (e.g., processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful.

No new application class has been emerging with the rise of Cloud Computing, but several application classes benefit from Cloud Computing properties:

- Mobile interactive applications. Such services will be attracted to the Cloud not only because they must be highly available, but also because these services generally rely on large data sets that are most conveniently hosted in large data centers.
- Parallel batch processing. Although thus far we have concentrated on using Cloud Computing for interactive SaaS, Cloud Computing presents a unique opportunity for batch-processing and analytics jobs that process terabytes of data and can take hours to finish. If there is enough data parallelism in the application, users can take advantage of the Cloud's new "cost associativity": using hundreds of computers for a short time costs the same as using a few computers for a long time. A special case of compute-intensive batch processing is business analytics.
- Extension of compute-intensive desktop applications. For example, the latest versions of the mathematics software packages Matlab and Mathematica are capable of using Cloud Computing to perform expensive evaluations.

An interesting field of application for Cloud Computing, and specifically for storage services, is disaster recovery. With major disasters happening, the classical situation of two datacenters one backing up the other could fail, as reported in [6, 7], describing a fail happened due to Hurricane Katrina, in 2005. The depicted situation consists of two data centers distant 60 miles. This distance should typically cover any disaster, but not all disasters are predictable. In fact, the hurricane made both data centers end up under 20 feet of water. Using virtualization and Cloud Computing, it is possible to geographically distribute data all over the world, and cut down data replication costs.

Therefore, even though Amazon's pay-as-you-go pricing (for example) could be more expensive than buying and depreciating a comparable server over the same period, the cost is outweighed by the extremely important Cloud Computing economic benefits of elasticity and transference of risk: not only disaster risks, but also the risks of over-provisioning (under-utilization) and under-provisioning (saturation). The key observation is that Cloud Computing's ability to add or remove resources at a fine grain (one server at a time with EC2), and within 2-5 minutes rather than weeks, allows resources to match workload much more closely.

From the Cloud provider's view, the construction of very large data centers using commodity computing, storage, and networking, uncovered the possibility of selling the resources on a pay-as-you-go model at a lower cost than many medium-sized data centers, while making a profit by statistically multiplexing among a large group of customers. From the Cloud user's view, it would be as startling for a new software startup to build its own data center as it would for a hardware startup to build its own fabrication line. In addition to startups, many other established organizations take advantage of the elasticity of Cloud Computing regularly, including agencies like NASA, newspapers like the Washington Post, movie companies like Pixar, and universities like Berkeley and Harvard.

7 Software Solutions

So far the cloud computing paradigm has been introduced, its main features have been analyzed, and the advantages it carries have just been explained. It's time to give a more detailed view to single systems. Cloud Computing systems are many, we can list Amazon Web Services, Google App Engine, Eucalyptus Systems, Ubuntu Enterprise Cloud, Microsoft Azure, SalesForce, Joyent, AppNexus, GoGrid, The Rackspace Cloud, and more.

We chose to give more details of some notable examples of Cloud Computing systems: a IaaS System, Amazon Web Services, a PaaS System, Google App Engine, and a Private/Hybrid Cloud System, Eucalyptus.

7.1 Amazon Web Services

Amazon Web Services (AWS) delivers a set of services that together form a reliable, scalable, and inexpensive computing platform "in the cloud". [1]

Amazon Web Services is a collection of services useful to work in the Amazon Cloud. The main services are:

- EC2 - Elastic Compute Cloud
- ELB - Elastic Load Balancing
- S3 - Simple Storage Service
- EBS - Elastic Block Storage

Elastic Compute Cloud (Amazon EC2) provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon EC2s web service interface (Figure 6) allows to obtain and configure capacity. It provides programmers with complete control of computing resources.

7.1.1 Amazon EC2

Amazon Elastic Compute Cloud presents a virtual computing environment, allowing to use web service interfaces to launch instances with a variety of operating systems, both default configurations or letting

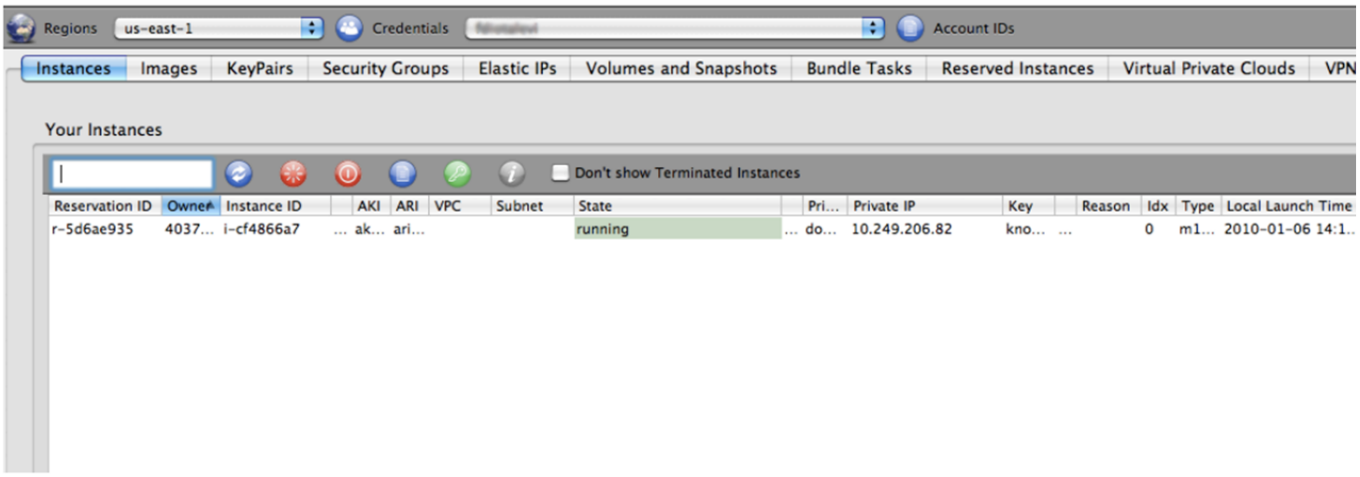

Figure 6. Elasticfox, AWS plugin for Firefox.

users customizing them, for example creating particular linux distribution configurations, load them in the application environment, manage access permissions, and run virtual system image using one or more instances at the same time.

To use Amazon EC2, one must follow these steps:

1. Select a pre-configured, templated image to get up and running immediately. Or create an Amazon Machine Image (AMI) containing applications, libraries, data, and associated configuration settings. The operating systems currently available to use with Amazon EC2 instances include Red Hat Enterprise Linux, Windows Server 2003/2008, Oracle Enterprise Linux, OpenSolaris, openSUSE Linux, Ubuntu Linux, Fedora, Gentoo Linux, Debian.

2. Configure security and network access on Amazon EC2 instance.

3. Choose instance type(s) and operating system, then start, terminate, and monitor as many AMI instances as needed, using the web service APIs or the variety of management tools provided.

4. Determine if running in multiple locations, utilizing static IP endpoints, or attaching persistent block storage to instances.

5. Pay only for the actually consumed resources, like instance-hours or data transfer.

Main EC2 features are:

Elastic Amazon EC2 enables applications to increase or decrease capacity within minutes. It is possible to commission even thousands of server instances simultaneously. Because this is all controlled with web service APIs, applications can automatically scale up and down depending on their needs.

Completely Controlled Administrators have root access to each owned instance. It is possible to stop an instance while retaining the data on a boot partition and then subsequently restart the same instance using web service APIs.

Reliable Amazon EC2 offers a highly reliable environment where replacement instances can be rapidly and predictably commissioned. The Amazon EC2 Service Level Agreement commitment is 99.95% monthly availability for each Amazon EC2 Region.

Auto Scaling Auto Scaling allows to automatically scale Amazon EC2 capacity up or down according to conditions defined by AWS users. With Auto Scaling, the number of used Amazon EC2 instances scales up seamlessly during demand spikes to maintain performance, and scales down automatically during demand lulls to minimize costs.

Assigned instances are priced basing on the offered compute power. The standard unit is EC2 Compute Unit (ECU), one EC2 Compute Unit (ECU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. Several configurations are possible, among them there are

- High-Memory Instances, which provide up to 68.4 GB of memory, 26 EC2 Compute Units (8 virtual cores with 3.25 EC2 Compute Units each), 1690 GB of local instance storage, 64-bit platform
- High-CPU Instances, which provide up to 7 GB of memory, 20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each), 1690 GB of local instance storage, 64-bit platform

7.1.2 Amazon ELB

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables to achieve greater fault tolerance in cloud applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.

7.1.3 Amazon S3

Amazon Simple Storage Service (Amazon S3) is storage for the Internet. It provides a web services interface that can be used to store and retrieve data on the web. Amazon S3 is built with a minimal feature set.

- Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each.
- Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- Reliability backed with the Amazon S3 Service Level Agreement. (99.9% monthly uptime percentage)

7.1.4 Amazon EBS

Amazon Elastic Block Store Amazon Elastic Block Storage (EBS) offers persistent storage for Amazon EC2 instances. Amazon EBS volumes provide off-instance storage that persists independently from the life of an instance. Amazon EBS volumes offer improved durability over local Amazon EC2 instance stores, as Amazon EBS volumes are automatically replicated on the backend. Amazon EBS provides the ability to create point-in-time consistent snapshots of your volumes that are then stored in Amazon S3, and automatically replicated across multiple Availability Zones.

7.2 Eucalyptus

EUCALYPTUS - Elastic Utility Computing Architecture Linking Your Programs To Useful Systems - is an open source software infrastructure for implementing on-premise clouds on existing Enterprise IT and service provider infrastructure[10]. Eucalyptus enables hybrid cloud and private cloud deployments for enterprise data centers. To enable the creation of a hybrid cloud, that is, a private cloud with the possibility to share the load with a public cloud, Eucalyptus supports the popular AWS cloud interface (see figure 7) allowing these on-premise clouds to interact with public clouds using a common programming interface. Along with managing virtual machines, the technology supports secure virtualization of the network as well as the storage infrastructure within the cloud environment. Eucalyptus is compatible with and packaged for multiple distributions of Linux including Ubuntu, RHEL, OpenSuse, Debian, Fedora, and CentOS.

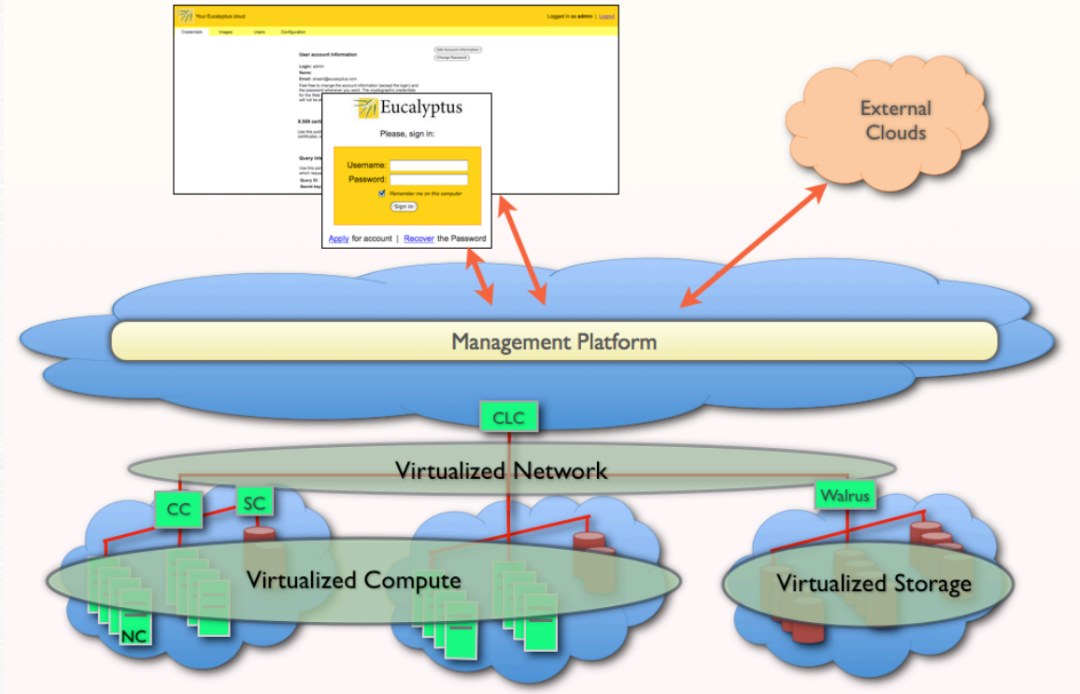

Figure 7. Conceptual Representation of the Eucalyptus Cloud. CLC is the Cloud Controller which virtualizes the underlying resources (servers, storage, and network). The Storage Controller (SC) provides block storage service (similar to Amazon EBS) while the Walrus storage system spans the entire cloud and is similar to the Amazon S3 in functionality. A Management Platform provides a one-stop console for the cloud administrator to configure and manage the cloud.

7.3 Google App Engine

Google App Engine is a platform for developing and hosting web applications in Google-managed data centers. Compared to other scalable hosting services such as Amazon EC2, App Engine provides more infrastructure to make it easy to write scalable applications, but can only run a limited range of applications designed for that infrastructure.

- App Engine can only execute code called from an HTTP request (except for scheduled background tasks).
- Java applications may only use a subset (The JRE Class White List) of the classes from the JRE standard edition.
- Java applications cannot create new threads.
- Java applications cannot create sockets.

Comparing to other systems, like Amazon AWS, Google App Engine restricts applications possibilities, but encloses the applications in a sandbox, avoiding programmers from interacting with operating systems, and providing applications with immediate scaling and load balancing.

Google App Engine is free up to a certain level of used resources. Fees are charged for additional storage, bandwidth, or CPU cycles required by the application:

- 1,300,000 HTTP requests
- 1Gb bandwidth
- 6.5 CPU hours
- 1Gb storage
- 7000 outgoing mails
- 100,000 tasks

These free quotas, provided by Google App Engine, make GAE a perfect choice for Small and Middle Businesses to startup, because allow SMBs to completely avoid hardware capital expenditure, and lowering maintenance expenditure.

A Google App Engine weakness is the absence of a Service Level Agreement, protecting cloud users from platform's outages.

8 Conclusions

Cloud Computing is a paradigm useful to small and medium businesses, by cutting down fixed costs of buying hardware, making "start-up" and growth easier. It is useful to industrial and academic research, too, allowing the execution of complex tasks in few time.

The advantages brought by Cloud Computing are many. Among them the illusion of infinite computing resources available on demand; the elimination of an up-front commitment by Cloud users, allowing companies to start small and increase resources only when needed; the ability to pay for use of computing resources on a short-term basis as needed and release them as needed.

Cloud Computing benefits span over several application classes. We can list mobile interactive applications, because they must be highly available, and these services generally rely on large data sets that are most conveniently hosted in large datacenters; parallel batch processing, because if there is enough data parallelism in an application, users can use hundreds of computers for a short time costs the same as using a few computers for a long time; extension of compute-intensive desktop applications, the latest versions of the mathematics software packages Matlab and Mathematica are capable of using Cloud Computing to perform expensive evaluations; disaster recovery, using virtualization and Cloud Computing, it is possible to geographically distribute data all over the world, and cut down data replication costs.

Two economic consideration can be made about Cloud Computing: even though a pay-as-you-go pricing could be more expensive than buying and depreciating a comparable server over the same period, the cost is outweighed by the extremely important Cloud Computing economic benefits of elasticity and transference of risk, besides disasters risks, the risks of overprovisioning and underprovisioning. From the cloud provider's view, the construction of very large datacenters at low cost sites using commodity computing, storage, and networking uncovered the possibility of selling those resources on a pay-as-you-go model below the costs of many medium-sized datacenters, while making a profit by statistically multiplexing among a large group of customers. In addition to startups, many other established organizations take advantage of the elasticity of Cloud Computing regularly.

There are three main factors contributing to the surge and interests in Cloud Computing: rapid decrease in hardware cost and increase in computing power and storage capacity, and the advent of multi-core architecture and modern supercomputers consisting of hundreds of thousands of cores; the exponentially growing data size in scientific instrumentation/simulation and Internet publishing and archiving; and the wide-spread adoption of Services Computing and Web 2.0 applications.

In the other hand, several difficulties must be overcome for Cloud Computing to be used on a large scale, the first one is the standardization of services offered by Cloud vendors; another obstacle is the limited support to relational database, offered by current cloud solutions; the latter difficulty to be overcome is the privacy of data located in a cloud. Once all of these difficulties will be surmounted, Cloud Computing will have the possibilities to be a massively used paradigm.

References

[1] Amazon. Amazon web services. http://aws.amazon.com.

[2] Apache. Hadoop, 2008. http://hadoop.apache.org/core/.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[4] J. Brodkin. Gartner: Seven cloud-computing security risks. *Network World*, 2008. http://www.networkworld.com/news/2008/070208-cloud.html.

[5] R. Chellappa. Intermediaries in cloud-computing: A new computing paradigm. In *INFORMS Dallas 1997 Cluster: Electronic Commerce*, Dallas, Texas, 1997.

[6] M. Creeger. Cto roundtable on virtualization. *Communication of the ACM*, 51(11):47–53, 2008.

[7] M. Creeger. Cto roundtable on virtualization, part ii. *Communication of the ACM*, 51(12):43–49, 2008.

[8] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI ' 04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA*, 2004.

[9] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, 2007.

[10] I. Eucalyptus Systems. Eucalyptus open-source cloud computing infrastructure - an overview, 2009. https://dspace.ist.utl.pt/bitstream/2295/584877/1/EucalyptusWhitepaperAug2009.pdf.

[11] A. Forestiero, C. Mastroianni, and M. Meo. Self-chord: a bio-inspired algorithm for structured p2p systems. In *CCGrid 2009 - IEEE International Symposium on Cluster Computing and the Grid*, 2009.

[12] I. Foster. What is the grid? a three point checklist. *GRID TODAY*, 2002. http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf.

[13] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10.

[14] J. Geelan. Twenty-one experts define cloud computing. *SYS-CON Media*, 2009. http://cloudcomputing.sys-con.com/node/612375.

[15] IBM. Autonomic computing manifesto, 2001. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.

[16] N. Leavitt. Is cloud computing really ready for prime time? *Computer*, 42(1):15–20, 2009.

[17] S. Lohr. Google and ibm join in cloud computing research. *New York Times*, 2007.

[18] J. Markoff. Internet critic takes on microsoft. *New York Times*, 2001.

[19] J. McCarthy. Mit centennial, 1961.

[20] M. Sheehan. Cloud computing expo: Introducing the cloud pyramid. *SYS-CON Media*, 2008. http://cloudcomputing.sys-con.com/node/609938.

[21] R. Stoica, D. Morris, M. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *In Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM'01*, 2001.

[22] T. Von Eicken. The three levels of cloud computing. *SYS-CON Media*, 2008. http://virtualization.sys-con.com/node/581961.

Index