Juha Rouvinen

Peeking inside the Cloud

Master's Thesis
in Information Technology
May 24, 2013

University of Jyväskylä

Department of Mathematical Information Technology

Jyväskylä

Author: Juha Rouvinen
Contact information: juha.p.rouvinen@student.jyu.fi
Title: Peeking inside the Cloud
Työn nimi: Kurkistus Pilven sisään
Project: Master's Thesis in Information Technology
Page count: 75
Abstract: Cloud computing is a relatively new computing paradigm that has received a lot of hype. Despite all the focus on clouds, there remains confusion about the exact nature of cloud computing. In this paper we take a look at various definition of cloud computing and look for the core features and characteristic of it. We find that cloud computing is at its core a combination of Software as a Service (SaaS) and utility computing, and its main characteristics are the infinite abstracted resources available on-demand and the instant and automatic scalability it offers. A comparison is made between cloud computing and various other related computing paradigms to find similarities and differences between them. We conclude that cloud computing does stand out on its own as a new computing paradigm.
Suomenkielinen tiivistelmä: Pilvilaskenta on uusi laskentamalli, joka on ollut suuren kiinnostuksen kohteena. Kaikesta saamastaan huomiosta huolimatta pilvilaskennan tarkasta määrityksestä on olemassa epäselvyyttä. Tässä tutkielmassa käymme läpi erilaisia alan asiantuntijoiden ja tutkijoiden antamia määritelmiä pilvilaskennasta ja pyrimme näiden pohjalta löytämään sen pääominaisuudet. Huomaamme, että pilvilaskenta on pohjimmiltaan SaaS:in ja utility computing:in yhdistelmä ja sen ominaispiirteet ovat sen rajattomat abstraktit resurssit (jotka ovat käytettävissä milloin tahansa, missä tahansa) ja sen välitön ja automaattinen skaalautuvuus. Vertaamme myös pilvilaskentaa muihin laskentamalleihin, jotka tavalla tai toisella liittyvät siihen. Tarkoituksena on löytää niiden yhtäläisyyksiä ja eroja. Päädymme johtopäätökseen, että pilvilaskenta "ansaitsee" olla oma laskentamallinsa.
Keywords: Cloud, Cloud Computing, Utility Computing, Software as a Service, Infrastructure as a Service, Grid Computing, Edge Computing
Avainsanat: Pilvi, Pilvilaskenta, Utility Computing, SaaS, IaaS, Grid-laskenta, Edgelaskenta

Contents

1 Introduction

Cloud, cloud computing, cloud services. The cloud seems to be on everyone's mind when talking about the future of IT. It carries a feel of innovation and endless possibilities. But what exactly is cloud computing? Is it something completely new, or just a fancy buzzword used to describe the movement of software, and more importantly hardware, into the confines of third-party operated massive data centers (the clouds)? We set out to gain a better understanding of the cloud computing paradigm and its effects on the industry. The goal is to give readers who might (or might not) be somewhat familiar with the cloud computing concept a solid, extensive and easy to understand look at this emerging new computing paradigm.

In this paper we take a closer look at what exactly makes up the cloud computing paradigm. We start with a brief look at the history and events that lead to the emergence of the clouds. Then we go over different existing definitions of cloud computing, and try to find out the core features and characteristics of cloud computing through them. We briefly cover the different main cloud types (*Public, Private, Hybrid*) plus a few others (*Community, Federated, Virtual Private*) that have been suggested by some sources.

We also take a look at cloud computing from the all the different main actors' (*Infrastructure providers, Service providers, Service users*) point of view. Basing the overview on the three main actors will hopefully make the effects, motivations, benefits, challenges and even worries of cloud computing easier to understand. The different cloud computing service levels (*Infrastructure as a Service, Platform as a Service, Software as a Service*) will be introduced and covered as well.

Finally we go over the many different terms, technologies and paradigms that have been associated with cloud computing and clouds. We make a side-by-side comparison between each of them and cloud computing, and attempt to find the similarities and differences between them. How do these technologies and paradigms manifest themselves in cloud computing? Does cloud computing fully embrace them? Has cloud computing changed them in some ways? The goal is to gain a better understanding of all the different terms being used in discussions about cloud computing, and how they related to it. By going over all these different concepts we also try to find out if cloud computing truly is a unique computing model, able to stand on its own. What are the core features that make it completely unique to any

of the other models?

These are the specific research questions we set out to find answers for in this paper:

1. What is cloud computing? What are clouds? What are the essential characteristics of them?

2. Why is there confusion about the definition of cloud computing? Is the concept too confusing or broad that this is warranted? Should the definition be redefined or adjusted to make it clearer?

3. What are the motives, benefits, concerns and challenges for all the actors in the cloud computing scenario?

4. What are the differences and similarities between the variety of technologies, computing models and paradigms that are often associated with cloud computing? How exactly are they related to clouds?

Based on the answers to these four questions, we try to answer one final question:

5. Can cloud computing be considered an independent new paradigm, or is it just a combination of existing approaches, concepts and technologies? Which of the features make it stand out from the other computing models?

2 Cloud Computing overview

We start with a brief look at the background of what has lead to the rise of cloud computing. What technological advancements and reasons are behind the emergence of clouds. How did it all start? After the background, we look at some of the definitions that the industry and researchers have given to cloud computing and clouds. As the cloud paradigm is still a fairly new concept and is made up of many different technologies and paradigms, there is no single textbook definition for it. At least not one everyone would agree on. From these definitions we attempt to find the core features and characteristics of cloud computing, and find out the ones that make it unique from other computing models. We also try to cover why there is confusion about the cloud computing paradigm, and if the definition should be adjusted and / or redefined to make it clearer. The chapter ends with a look at the different cloud types that have been identified.

2.1 Movement into the clouds

The cloud computing model can be tracked back to large Internet based companies building their own infrastructure [28]. Google, Amazon, IBM and other Internet giants have large scale server farms spread around the world, powering their operations. These massively scaled and distributed farms are at the core of their business, this is where they specialize in. Cloud computing started as a business idea to these companies: Why not scale up these data centers to support third-party use [34]? Allow customers to make use of these resources, whether they be raw computing power, storage capacity, networking, programming and collaboration tools, applications or services [28], and determine the price based on the resource consumption. So cloud computing did not start out as a strategy to build massive data centers and sell their resources as utility computing, rather these data centers were already being built in the early 2000's to support the massive growth of web services [2]. The concept and potential of cloud computing was only later realised, when the infrastructure to support it had already been built.

The first to test this new idea was Amazon with their Elastic Compute Cloud (EC2) back in October 2007 [34]. Making use of virtualization, the customer can create a complete software environment, after which a machine instance is created

to run it. This instance can be configured to have more memory, more cores, more storage, and the customer can create and destroy these instances at will [34]. This allows the service to scale to whatever resources it needs, at any given time (thus scalability being one of the key features of cloud computing). And as mentioned before, the customer only pays for the resources used, removing the over- and under-provisioning scenarios that are bound to happen if the company had the software running on their own servers [34].

As Brian Hayes notes in his article [17], cloud computing is in a sense similar to what computing centers were 50 years ago. Users with terminals connected to the central computer over telephone lines to have their computation done. The arrival of personal computers in the 80s was to "liberate" computing from the centralized control over to the individual users. In the cloud computing model we are seeing a reversal of these roles again [34]. The dramatic growth of affordable high-bandwidth networking in North America, Europe and Asia has been critical in making the shift to the clouds possible [28]. Quite interestingly computing pioneer John McCarthy predicted, way back in 1961, that "computation may someday be organized as a public utility" [9]. Though this was probably just an ideal of expanding access to the computing centers back then to the masses, instead of just selected organizations and people. Still, it does show the connection between the computing centers of the old days and cloud computing, and how we are actually going backwards towards the 50 year old computing model. Just the underlying technologies, business motivations and scale of things have changed.

Though it still remains to be seen how far and wide can the cloud computing model grow. Can it ever completely replace the personal computers, like some of the wildest theories imagine? Or will they continue to coexist and supplement each other? Some [34], [37], [38] have presented futuristic views of an ultralight input output device with a screen that does all the computations in a cloud. Clearly this sort of complete change is not going to happen any time soon, not because of physical limitations like Internet connection speeds and availability around the world, but also because many companies have built their businesses around purchasable, locally run applications [34]. There are even non-hardware related issues like trust and privacy in having all your personal data somewhere out there in an undefined "cloud" [34]. Foster et al. [9] believe that cloud computing and client computing will continue to coexist and evolve together, due to people and organizations not trusting clouds with mission-critical applications and sensitive data, and due to network related problems like availability and reliability of the cloud. The role of data and data management (mapping, partitioning, querying, movement, caching, repli-

cation, etc) will continue to grow and become more important for both cloud and client computing as more and more applications become data-intensive. The location of data and exploiting data locality is important as the movement of data to distant CPUs is becoming the bottleneck for performance and increasing costs quickly [2], [6], [9].

One big concern and a topic that is currently under much discussion and research is the data security and privacy risks of cloud computing [6], [9], [17], [20], [29], [31], [37], [38]. It could even be the single greatest fear that organizations have about cloud computing [6], [29]. Grobauer et al. [15] note that often the discussion about cloud computing security isn't well defined, as the terms *risk*, *threat* and *vulnerability* are used interchangeably, and that not every issue raised in these discussions is specific to just cloud computing. They point out that the focus in the security discussions about cloud computing should be on the *vulnerabilities* that it can make more significant, and the new vulnerabilities that it can introduce. We will cover these security concerns more closely in the latter half of this chapter.

2.2 Definitions

2.2.1 Cloud Computing

In their paper *A Break in the Clouds: Towards a Cloud Definition*, Vaquero et al. [33] note that there has been confusion about the overall picture of what cloud computing really is. They further note that the variety of different technologies in cloud computing, some of which are not new at all (virtualization, utility computing, distributed computing) make the overall picture confusing, and that cloud computing, as a new concept, suffers from early stages of hype. This turns the cloud into an excessively general term that includes almost any solution that allows the outsourcing of all kinds of hosting and computing resources [33]. Armbrust et al. [2] and Foster et al. [9] similarly note the confusion around the exact meaning of cloud computing. Vaquero et al. [33] emphasize the need to find a unified definition for cloud computing, which would benefit further research and businesses alike. To address this, they attempted to give cloud computing a complete definition in their paper. They studied over 20 definitions available at the time (2008), noting that many of them focused on just certain aspects of the technology. Their proposed definition was: *"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization.*

*This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs *(Service-Level Agreements)."*

When looking for a minimum common denominator in all the proposed definitions, they found the terms *scalability*, *pay-per-use utility model* and *virtualization* being closest fitting to this minimum definition. Other features that were also mentioned in some of the definitions include *user friendliness* (usability), *variety of resources* (versatility), *Internet centric* and *resource optimization* (high utilization rate).

Cloud computing is built around many existing technologies and architectures (centralized computing power, utility computing, distributed computing, software as a service, etc). Clouds are new in that they integrate all of these computing models together, and that the integration requires the computing power to be shifted from the processing unit to the network [14], [34], [37], [38].

Zhang et al. [38] list the characteristics of cloud computing as follows (the items marked with * are explained for easier comparison):

1. Ultra large-scale (* the clouds can have hundreds of thousands of servers)
2. Virtualization
3. High reliability
4. Versatility (* able to run any type of applications and software)
5. High extendibility (* in other words, high scalability)
6. On demand service (* in other words, pay-per-use, utility computing)
7. Extremely inexpensive (* compared to having your own servers)

Yang et al. [37] define cloud computing with the following characteristics:

1. Virtualized pool of computing resources
2. Manage a variety of different workloads (* versatility)
3. Rapid deployment and increase of workload (* scalability)
4. Recovery from hardware / software failures (* reliability)
5. Real-time resource monitoring and rebalancing

Takabi et al. [29] and Dillon et al. [6] quote the US National Institute of Standards and Technology (NIST) definition of cloud computing in their papers: *"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. This cloud model promotes availability and is composed of five essential characteristics, three delivery models, and four deployment models."*

The five key characteristics from this definition are *on-demand self-service*, *ubiquitous network access*, *location-independent resource pooling*, *rapid elasticity* and *measured service* (resource usage is constantly metered, enabling the pay-per-use model) [6], [15], [29]. Grobauer et al. [15] state that the NIST's definition framework and essential characteristics list has evolved to be the de facto standard for defining cloud computing. Dillon et al. [6] note that NIST's definition seems to include key common elements widely used in the cloud computing community. The delivery models (or service levels) and deployment models (cloud types) from this definition will be discussed later in this chapter.

Foster et al. [9] offer their take on cloud computing "to the already saturated list of definitions":
"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."
They note that the key factors that differentiate it from traditional computing paradigms are:

1. The massive scalability
2. The encapsulation into an abstract entity that delivers different levels of service to customers outside the cloud
3. It is driven by economies of scale
4. The services can be dynamically configured (via virtualization or other means) and delivered on demand

In his article [21], Dave Malcolm lists five defining characteristics of cloud computing (mostly from the Infrastructure providers aspect) as follows:

1. Dynamic computing infrastructure
   - Virtualization, scalability and automatic provision creating high level of utilization and reuse of the infrastructure.

2. IT service-centric approach
   - Providing a business service for the customer, hiding or reducing the hardware, system and network administration.

3. Self-service based usage model
   - Give customer easy-to-use and intuitive way to upload, build, deploy, schedule, manage and report on their business services on demand.

4. Minimally or self-managed platform
   - Resources should be self-managed via software automation, such as a provisioning engine for deploying services, mechanisms for scheduling and reserving resource capacity, etc.

5. Consumption-based billing
   - Consumers only pay for resources they use (pay-per-use, utility computing). The system must be able to capture this usage information.

A TechPluto article [27] looks the picture from a slightly different view: What characteristics should an application have to be suited for cloud deployment? They list the following four characteristics:

1. Requires flexibility
   - The usage is inconsistent or spiky. The cloud's scalable and pay-per-use qualities remove over- and under-provisioning issues, and reduces costs.

2. Is growing exponentially or demands scalability
   - An application that is expected to or has potential to grow is a good candidate for cloud deployment so we don't need to take risks by building a too large or too small infrastructure to run it.

3. Wants to run economically
   - Only pay for used resources (pay-per-use, utility computing), reducing costs.

4. Independent in nature
   - An application that needs to regularly converse with other applications and databases residing elsewhere can hinder the benefits of cloud and create security risks. The requisite applications and databases should also be deployed into the cloud.

Armbrust et al. [2] similarly look at what kind of applications are particularly well suited for cloud computing. They note that services that must be highly available, mobile and rely on large data sets (possibly from different sources) are well suited for clouds. A service with a highly variable or unknown demand is very well suited for clouds, as the automatic scaling can lead to cost savings and reduced risks that would result from under- or over-provisioning. Another point they bring up is that cloud computing works well for batch-processing and analytic tasks that can take hours to finish. They note that using hundreds of computers for a short time costs the same as using a few computers for a long time. These types of tasks would be especially well suited to be done in the cloud if they aren't done regularly, since the task could be set up and finished quickly and easily without any investment in your own hardware. Overall they note that a good test is to compare the cost of computing in the cloud plus the cost of moving the data in and out of the cloud to the time saved from using the cloud. Particularly good tasks and applications for cloud computing are ones that have a high computations-to-bytes ratio, for example some symbolic mathematics and 3D image rendering [2].

Ranjan et al. [26] note that while cloud computing might not seem radically different to existing paradigms and technologies at the high-level, it does have several technical and nontechnical characteristics that differentiate it from the other models. The technical characteristics include *on-demand resource pooling*, *rapid elasticity*, *self-service*, *almost infinite scalability*, *end-to-end virtualization support* and *robust support of resource usage metering and billing*. The nontechnical differentiators include *pay-as-you-go-model*, *guaranteed SLAs*, *faster time to deployments*, *lower upfront costs*, *little to no maintenance overhead* and *environment friendliness* [26].

Rajan et al. [25] call cloud computing the fifth generation of computing, after *Mainframe*, *Personal Computer*, *Client-Server Computing* and the *Web*. They go on to list the advantages of cloud computing as faster, simpler and cheaper services, high elasticity, optimized utilization of computing resources, unlimited resources, service orientation, lower power consumption, high availability and scalability and no data loss.

Dong Xu [36] describes cloud computing as a usage model in which resources (such as hardware, software and applications) are delivered as scalable and on-demand services via public network in a multi-tenant environment. All the resources in the cloud are available as utility, and the cloud offers infinite scalability. Cloud is the resource network that provides all this functionality [36]. He further notes that the most interesting thing about cloud computing is not the technology, but rather the new evolving social standards and business models it brings about.

Gong et al. [14] also note the confusion around cloud computing definition. They quote Oracle CEO L. Ellison saying that cloud computing is nothing more than "everything that we currently do". They even question if a clear definition of cloud computing is important, as long as we understand its essential characteristics. They identify these characteristics as:

1. Service oriented (* utility computing)
2. Loose coupling (* user applications and data inside the cloud are independent of each other)
3. Strong fault tolerance (* due to the independence of applications and data)
4. Business model (* Clouds are geared towards commercial use, unlike Grids)
5. Ease of use (* usability)

Armbrust et al. [2] use the users perspective to define cloud computing. In their view, cloud computing can refer to both the applications being delivered as services over the Internet (software as a service), and the hardware / systems software in the data centers that provide these services (the cloud). They further call the act of offering the cloud to customers utility computing. They conclude that cloud computing is thus the sum of software as a service and utility computing. People can be users or providers of software as a service, or they can be users or providers of utility computing [2]. In these terms the providers of software as a service are also the users of utility computing. These actors will be discussed in more detail a bit later. Figure 2.1 [2] illustrates this user / provider division.

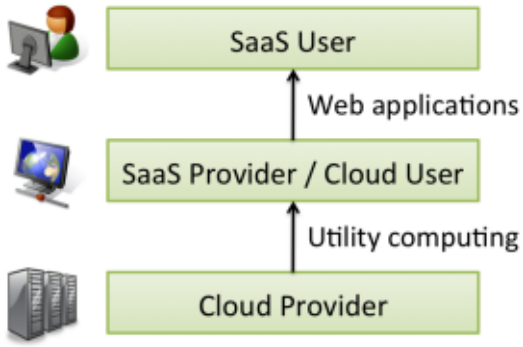

Figure 2.1: Users and Providers of Cloud Computing [2].

Following this definition of cloud computing by Armbrust et al. [2], one can understand why there is confusion about the concept of cloud computing. Since it refers to both the software as a service concept, which by itself is nothing new, and the fully realized utility computing, perhaps the term ”cloud computing” is simply too bloated. When we look at what really makes the clouds and cloud computing unique, most of these things happen at the utility computing side of this division. Personally I feel that the definition given by Armbrust et al. is a clear one and can give a better understanding of the cloud computing concept. When explaining cloud computing to someone not proficient in the area, one should focus on the utility computing side of things, and on the infrastructure / service provider interaction, not so much on the end user. Armbrust et al. actually note that this side of cloud computing has received less attention [2], despite being the ”new” thing.

Armbrust et al. [2] further note three things from the hardware point of view that are new in cloud computing:

1. The illusion of infinite computing resources available on demand (* scalability)
2. The elimination of an up-front commitment by cloud users (* pay-per-use)
3. The ability to pay for use of computing resources on a short-term basis as needed (* pay-per-use, utility computing)

Virtualization definitely seems to be a characteristic that appears in almost all papers, articles and definitions of cloud computing. Sun Microsystem’s Cloud Computing primer [28], call virtualization ”a cornerstone design technique for all cloud architectures”. It can refer to the virtualization used in the actual data centers to run multiple, independent operating instances simultaneously on a single server [34], and it can refer the abstraction of physical IT resources from the people and applications using them [28]. While virtualization is nothing new, it is a key component of what makes cloud computing work [9], [15], [16], [25], [28], [29], [33], [37], [38]. Though there is criticism about virtualization in clouds. Keller et al. [18] note that virtualization is only an implementation detail for clouds, rather than a key feature of clouds. Their criticism is aimed towards the security threats that can result from malicious attacks in the virtualization layer. They go on to propose getting rid of the virtualization layer altogether, and offer their NoHype architecture as an alternative that aims to retain the key features enabled by virtualization, while addressing the security issues.

As many have concluded [9], [14], [28], [29], [33], [34], [37] cloud computing is built around existing technologies. So what makes it so revolutionary? The other

words that keep popping up in cloud computing articles are scalability and utility computing. This is something new that cloud computing can truly offer. A small or medium sized company can not produce their own infrastructure that can be as scalable and cost-efficient as a cloud is. To fully realize the economies of scale a company needs to have extremely large data centers, which can be a hundred-million-dollar undertaking. Not only this, but a large-scale software infrastructure and operational expertise required to run it all are required [2].

The disappearance of companies own servers into the cloud allows the companies to use only the resources they need at a given time, not more, not less. This increases resource utilization rates dramatically, and allows the service to automatically scale up for peak demands (and afterwards scale down) without investment in new infrastructure, personnel or software [28]. What other technology before cloud computing has allowed these kinds of results, at this level of efficiency? Armbrust et al. call this elasticity of resources, and the affordable access to it, unprecedented in the history of IT [2].

Pay-per-use is another concept that was found in most of the cloud computing definitions. It is related to the utility computing model where computing resources are used as a service [14]. Armbrust et al. [2] point out the difference between pay-per-use and renting: Renting involves paying a price to acquire resources for some time period. It doesn't matter if those resources are used, you still pay the price. Pay-per-use (or pay-as-you-go) is charged based on actual use of the resources, independent of the time period over which those resourced were used. This is why the term renting is not appropriate to describe cloud computing (unless the cloud resources truly were offered in such a way).

While it is not an absolute requirement of clouds to be billed in a pay-for-used-resources way, it seems to be the dominant model used for now [2], [14], [21], [26], [33]. For example the Amazon EC2, Microsoft Windows Azure and Google App Engine use the pay-per-use model. The Amazon EC2 has different types of "on-demand instances" that are billed per hourly use. The prices can vary depending on specific types of instances (high-memory, high-CPU, High-I/O, etc), the type of operating system (Linux/UNIX, Windows), the location of the cloud (US East, US West, EU, etc). In addition to the on-demand instances, data transfer to in and out of the Amazon EC2 is billed separately [1].

Still, cloud computing is a fairly new technology and as it becomes bigger and more matured, new business models could appear to support it. Already more advanced payment models are being explored [2], [33]. Maybe subscription based models? Or some sort of package deals, much like telecommunication companies

offer to their users? Microsoft Windows Azure already offers discounts for 6 and 12 month commitment plans [35], although these are geared just for long term investments. The customer is still billed based on resource consumption. Even if newer, more sophisticated pricing models do appear, the usage-based pricing is likely to persist due to being simple and transparent [2].

Zhang et al. [38] call cloud computing "extremely inexpensive", and other sources state cost-reduction as one of the main motives for moving into the clouds [27], [28], [29]. This could hint that in the future cloud computing prices could go up, once the technology has matured and the user base has grown large enough. If the choice to move into the cloud is obvious, there could certainly be room for the prices to go up. The operating costs of the massive data centers that power the clouds are high, with the power and cooling costs alone taking up one third of the overall cost [2], which could also drive up the prices. A significant price increase would be unlikely to happen in a short time period, since it would require most if not all of the big players to increase the prices together, or the ones making the move first could suffer from consumers moving over to other cloud providers. Since cloud computing is still a fairly new business, a low cost level is probably used to get competitive edge over other cloud providers, and also to attract new customers to grow the user base. Possible cloud providers might want to act now, before a single (or a few) massive cloud providers take over the industry [2].

2.2.2 Public, Private and Hybrid Clouds

One thing to consider with clouds is the ownership and purpose of the cloud infrastructure. We can divide the clouds into three different types: Public, Private and Hybrid Clouds [6], [16], [25], [28], [37]. Figure 2.2 illustrates the different cloud types:

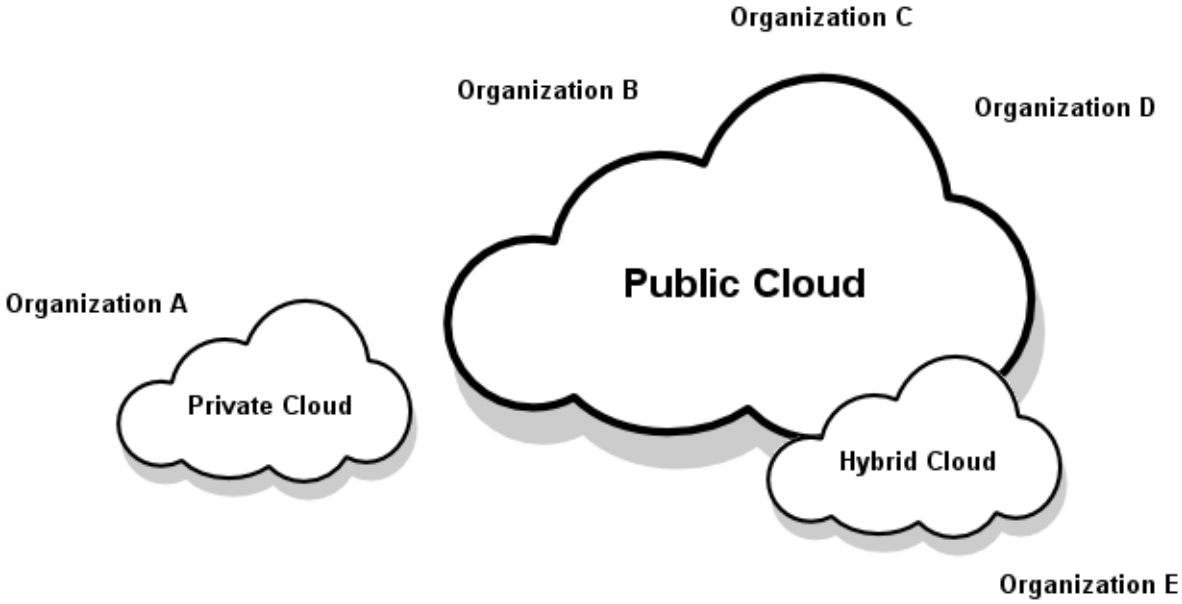

Figure 2.2: Cloud Types.

Most of what we have covered about clouds so far falls into the *Public Cloud* category. Public clouds are operated by external companies who offer their infrastructure and resources to be utilized by customers. Many different customers can have their software run in the same cloud, on the same resources, without ever being aware of each other. The fact that these public clouds are usually massive in scale [38], and are managed by companies who specialize in building these type of infrastructures [34], allows them to fully harness the scalability and efficient utilization rates associated with cloud computing. Public cloud offerings can raise issues with security, regulatory compliance and quality of service [37].

The next type of cloud is the *Private Cloud*. These are cloud infrastructures owned and managed by a single company, the distinction being that the cloud is not available for public use [2], [6], [25]. The private cloud could be intended to run a single, or preferably multiple applications owned and created by the cloud owner, their partners or whoever they decide to give access to the cloud. The private cloud gives the owner total control over the resources, and is good for companies dealing with data protection, privacy and service-level issues [6], [28], [37]. A private cloud can provide better availability and reliability of high priority applications and data, better security and allow self governance and control of the infrastructure [16]. Other reasons like optimizing the utilization of existing in-house resources, retaining full control of mission-critical activities and too high data transfer costs to public clouds can motivate a company to use a private cloud [6].

Though one has to ask, is this cloud computing anymore? One of the main innovations of cloud computing was getting rid of the hardware and all the costs and issues related to it, and to buy computing resources as a service (utility computing) [34]. If the company owns and operates the cloud infrastructure and the service(s) being run on it, isn't that what we have been doing for many years already? What about the scalability and efficiency of a private cloud? One of the main characteristics of clouds are their ultra large-scale [38]. How many companies can build, maintain and operate ultra large-scale server farms? The scalability, efficiency and cost advantages of the cloud come from the massive scale of the infrastructure, requiring tens of thousands of computers [2]. If company X has 1000 servers running some of their own services, will that be able to scale for peak demands? And if yes, how well and cost-effectively are the resources being utilized? Another problem with the private cloud concept is just how big does the "cloud" have to be to qualify as a cloud? If company Y has 100 servers running their services, is that a private cloud? Those 100 servers could be enough for their needs, but can we honestly call it a cloud? The private cloud concept clearly starts to lose one or more of the core qualities of cloud computing. Armbrust et al. do not even include private clouds in their view of the cloud computing concept [2], though their view of cloud computing is heavily based on utility computing (buying / selling the computing resources).

I can see this being the main issue causing confusion about cloud computing. If we accept the private cloud definition without any limitations, it automatically qualifies A LOT of online services as cloud computing. Maybe the scale and efficiency of the cloud could be used to separate real clouds from just ordinary data centers? One could argue that Facebook operates a private cloud, simply because their infrastructure does match ultra large-scale associated with clouds, and it is probably able to scale to meet peak demands fairly well. But it does not fill the utility computing aspect of buying computing resources as a service. And how well does it meet the efficiency aspect? Delivering a scalable infrastructure while at the same time retaining high utilization rate is extremely hard, if not impossible due to average workloads being much lower than peak workloads [2], [34]. A public cloud solves this issue for the service provider, but a private cloud does not.

A lot of services are titled or called clouds these days (it does have a nice ring to it), but should they really be called clouds? Does it even matter what they are called? Being able to call just any online service cloud or cloud computing could water down the concept of clouds and keep creating confusion about the technology. Perhaps we would need a more refined definition concerning private clouds and clouds in general, at least for academic use. One that concentrates on the aspects of

resource utilization efficiency, scalability and cost effectiveness.

The third and final cloud type is the *Hybrid Cloud*. This is basically a combination of the public and private clouds. Parts of the service are run on the company's own servers, and parts are outsourced to an external cloud, and this happens in a controlled way [6], [28]. The hybrid cloud could also be designed to outsource workloads to public clouds when a peak in usage occurs, and scale back as the extra computational resources are no longer required [37]. The hybrid cloud partly fills the utility computing model, and if designed correctly, can take advantage of the scalability and efficiency of "proper" massive-scale clouds. The problem can be in finding out how to effectively distribute the applications across the different environments [28]. As the TechPluto article [27] notes, an application suitable for cloud computing should be independent in nature, and the requisite applications and databases should all reside in the same cloud. Constant data exchange between the company's own servers and the cloud could create security risks, problems with bandwidth utilization [27] and problems for complex databases and synchronization [28]. The hybrid cloud concept has raised issues of cloud interoperability and standardization [6]. We will take a look at these issues later in this chapter.

If we look past the complexity and problems in designing an application that works well in a hybrid cloud, it does offer some interesting advantages. Having the core application running on local servers and being in control of the environment, while taking advantage of the cloud for scalability for peak-demands. It does leave some of the hardware issues and costs in the hands of the company, possibly reducing them depending on just how large is the local infrastructure. Thus it partly misses out on one of the main advantages of clouds. A truly hybrid cloud could even be run on multiple different clouds and locally, but this would likely just exacerbate the hybrid cloud design problems even further.

Some [6], [29], [37] have further introduce the *Community Cloud*, a joint effort where several organizations construct and share the same cloud infrastructure and the related policies, requirements, values and concerns. I think the community cloud is already presented by the three cloud types mentioned before. If only the community has access to the cloud, it could be considered a private cloud. An example of this could be an academic cloud, shared and used by a number of universities. If on the other hand the community cloud concept was a joint effort to build a publicly available cloud, it would fall under the public cloud category. If the cloud further relied on outsourcing some of its workload to other public clouds, it would be a hybrid cloud. Then again clouds are generally considered to be owned and managed by a single massive cloud provider, so perhaps the community cloud does

deserve to be mentioned as a special case of its own.

One interesting concept to think about is multiple massive clouds that help each other out by distributing work when one or more of the clouds can no longer handle the workload. A network of clouds, called a *Federated Cloud* [26], [33]. Such a network could be almost infinitely scalable. But it could also make the problems of hybrid clouds (security risks, synchronization, trust and legal issues, etc) even worse, and designing the cloud interaction and management effectively could prove to be a challenge. They would need to be united by a set of protocols and appropriate software [28]. Also a key challenge for the federated cloud would be to define a mechanism that ensures mutual benefits for all the individual clouds. Research into this area has already taken place, trying to apply market-oriented mechanisms to coordinate the sharing of resources [26].

Perhaps a simple model of cooperation that is not so much legally binding, but rather an alliance from which each involved cloud can benefit from: When a cloud is running out of resources, it can query other clouds in the federation, asking if they capable and willing to take on some of the work. The payment for such help could be defined in the terms of the federation, or participating clouds in the federation could even make competing offers for the help. This negotiation system could be automatic, each cloud determining their ability to take more work on the current utilization rate, expected workloads and other factors, some of which could be manually configured as needed. The cloud asking for help would determine which ”offer(s)” seem most beneficial for the required work. The Amazon EC2 has this sort of dynamic bidding in the form of ”Spot Instances”. These instances make use of the unused EC2 capacity, and the price fluctuates depending on the supply of and demand for the spot instances [1].

Figure 2.3 illustrates the Community Cloud and Federated Cloud concepts:

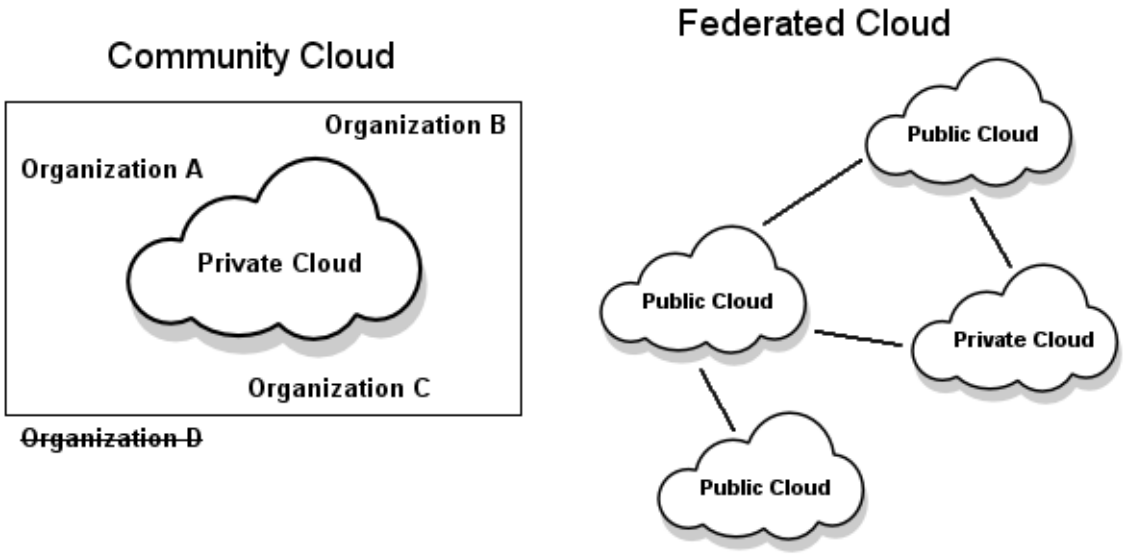

Figure 2.3: A private Community Cloud and a Federated Cloud. The rectangle around the community cloud represents the closed nature of the system. Note that the federated cloud can be comprised of both public and private clouds.

Dillon et al. [6] introduce one more cloud type in their paper, the *Virtual Private Cloud*. It has been introduced as part of the Amazon Web Services (AWS) platform. It offers a secure and seamless bridge between a company's own IT infrastructure and the Amazon public cloud. The virtual private cloud is a mix between a public and a private cloud. It qualifies as a public cloud since it uses the same computing resources that Amazon has pooled for the general public. However the connection between the company's own infrastructure and the cloud is secured through a virtual private network, AWS dedicates "isolated" resources for the virtual private cloud, and all the company's security policies are applied on the resources in the cloud. These give it the security and control advantages of a private cloud, while also giving it the flexibility advantages of a public cloud [6].

3 Main actors

So what are all the different parties involved in cloud computing? We already covered how Internet giants like Google and Amazon saw a new business model in cloud computing. Vaquero et al. [33] attempted to distinguish the kind of systems where clouds are used and the different actors involved in them. They identified three main actors, the *Infrastructure providers*, *Service providers* and *Service users*. Service providers create Internet based services for the service users. Infrastructure providers (Google, Amazon, etc) provide the servers and other infrastructure for the service providers to run their applications and software on. We will use these actors as the basis to explain cloud computing, from each of their perspectives. The goal is to find the motivations, advantages, disadvantages, challenges and overall effects that cloud computing presents to each actor. Figure 3.1 [33] illustrates the different actors in the cloud computing scenario:

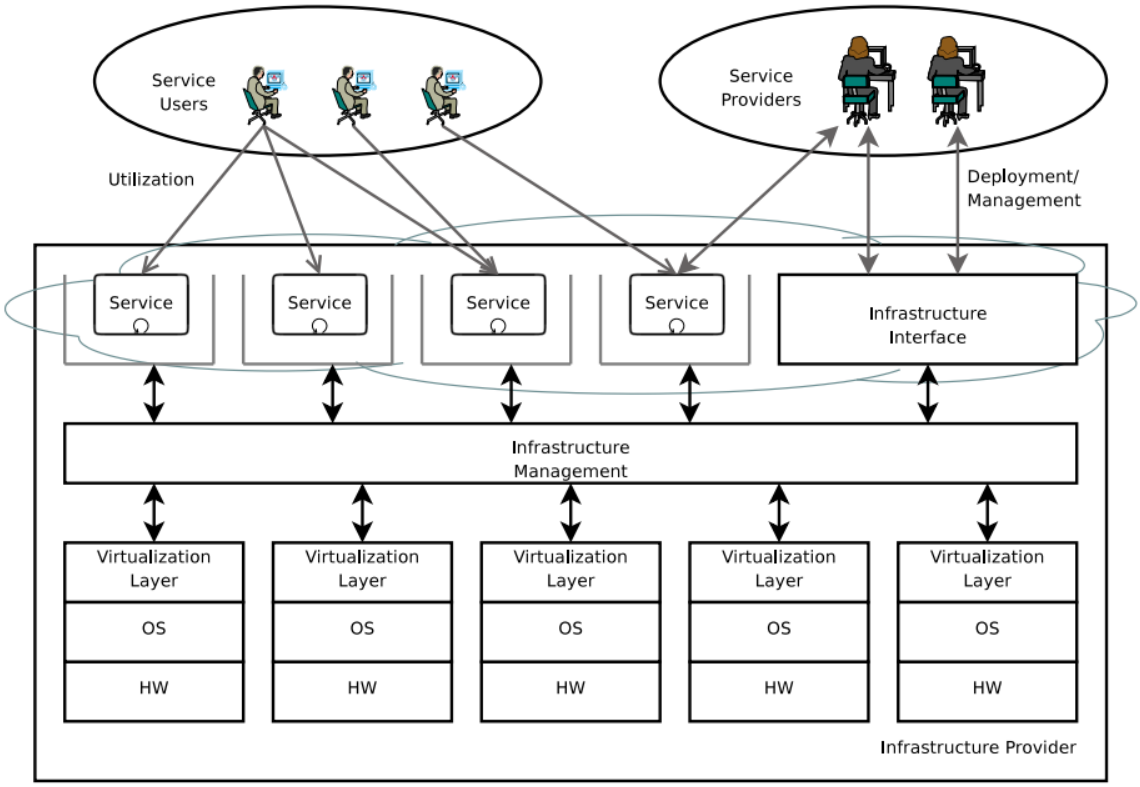

Figure 3.1: Cloud Actors [33].

There are three main service levels (or delivery models) identified and largely agreed upon in cloud computing: *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)* and *Software as a Service (SaaS)* [6], [9], [10], [14], [15], [16], [25], [28], [29], [31], [32], [33], [36], [37]. Though there is some criticism against this clear cut definition [2], stating that the differences among all the "X as a Service" can be hard to distinguish. Other service levels have also been suggested, such as *Data storage as a Service* [6], [36], *Hardware as a Service* [37], *Desktop as a Service* and *Backend as a Service*. In this paper we will be using the infrastructure-, platform- and software as a service division.

We've already covered the Infrastructure as a Service (IaaS) scenario: Infrastructure providers give service providers access to their computing resources, such as storage, processing capacity and network, and charge based on the resources used. These resources can be pooled to handle any type of workloads, from batch processing to server/storage augmentation during peak loads [28]. Perhaps the most well known examples of IaaS are the Amazon Elastic Compute Cloud (EC2) mentioned earlier, and the Amazon Simple Storage Service (S3) that focuses on storing and retrieving data from anywhere on the web.

Platform as a Service (PaaS) offers a complete development environment and a software platform in which to develop and run applications and services. The development environment often contains a solution stack, for example a Linux distro, a web server and a programming environment such as Pearl or Ruby [28]. It can also provide tools and services to support all phases of software development and testing, such as application design, database integration, security, version management and collaboration tools. The hardware resources required to run the development environment and the hosted applications and services are automatically scaled to meet the needs, in a transparent manner [33]. A PaaS can simplify and hasten software development and testing by providing the developer with a complete hardware / software package, removing the need for the developer to acquire, install and manage these assets themselves. While a PaaS can provide a great deal of flexibility, it may be constrained by the capabilities and software offering of the service provider [28].

A popular example of PaaS is Microsoft's Windows Azure, which offers development on multiple operating systems, tools, frameworks and languages (e.g. Java, .Net, PHP, Python). Another example is the Google App Engine, which is a platform for developing and hosting web applications. It currently offers support for Java and Python languages and frameworks. These development platforms are usually billed in the pay-per-use fashion, much like the cloud infrastructures are. They can

be charged for used resources like storage and bandwidth, but also for additional services like technical support [35].

The final scenario, Software as a Service (SaaS), is where the customer is provided access to a service or an application that is hosted in a cloud infrastructure (or a data center). This scenario is naturally highest in the abstraction level. The customer does not need to know anything about the cloud system powering the service or application, or even be aware that it is being run in a cloud. Basically, SaaS is just like any normal application, just not run locally but in the cloud. This removes the need to install and update the software locally, simplifying things for the user. It can also reduce the total cost of a software if it's billed in a pay-per-use pattern instead of a large upfront cost [14]. It can provide more flexibility for testing new software or using the software for only a limited time, but this of course depends on the terms of use.

The customer typically has access to the application or service through a thin client, which often runs in a web browser to make it more accessible [20]. Being tied to a web browser can put some limitations on the user interface. There are several initiatives to create a more versatile and richer user experience for web applications, for example the eyeOS "web desktop" that acts like an operating system inside a web browser, or the Adobe Integrated Runtime (AIR) application that bypasses the browser altogether [17]. The AJAX technologies and the new HTML5 revision also aim to enrich the browser based user experience. Cloud services can take the best of both worlds and offer multiple ways to access them, for example the Microsoft SkyDrive cloud storage offers a HTML5 based web interface, as well as a separate desktop application.

SaaS offerings can vary from typical business-to-business services such as accounting, sales, marketing, collaboration and management applications to mass market applications like web-based office suits and e-mail services, like the ones offered by the Google Apps service. The billing methods for SaaS aren't necessarily tied to the resource consumption in the cloud (as is often the case with IaaS and PaaS [2], [14], [21], [26], [33]), as it is completely invisible to the end user [14]. Many of these services can even be free to use [14], for example Google Gmail and Microsoft SkyDrive are free to use (with a limited storage). Additional storage space can be rented for a cost.

3.1 Infrastructure providers

These are the companies who host the servers and other infrastructure that is needed to power the cloud applications. The offering can range from raw computing power, storage capacity and networking to virtually any IT resource. For the Infrastructure providers cloud computing offers the business prospect of customers paying to use these resources. A good example of this is Amazon's Elastic Compute Cloud (EC2) that we have already covered earlier. The Infrastructure providers can operate on the other levels of cloud computing as well [28], [36]. For example Google has the Google App Engine (PaaS) and the Google Docs web-based office suit (SaaS) running on their own cloud infrastructure. Another example is the Microsoft's Windows Azure. It offers developers a software environment and tools to develop applications online (PaaS), and then allows the developers to host the applications in the cloud (IaaS). Both of these services are billed separately on the pay-per-use rate. Microsoft also offers the Office 365 web-based office services (SaaS), thus operating on all the three levels of cloud computing. Salesforce.com started out with a SaaS offering, but has since expanded to PaaS level with their Force.com development platform.

So who should (or more accurately, *can*) become a cloud infrastructure provider? What benefits could be gained from it? As we concluded before, for a company to become a cloud infrastructure provider, it must have very large data centers and software infrastructure, and the operational expertise required to run them [2]. Armbrust et al [2] list 6 reasons that might motivate a company to become a cloud infrastructure provider:

1. **Make a lot of money**. This is made possible by the extremely large size of the data center. Bulk purchases of hardware, network bandwidth and power allow the company to realise the economies of scale and operate well below the costs of medium sized data centers. This allows them to offer the cloud infrastructure to customers at a very attractive and competitive price.

2. **Leverage existing investment**. For a company that already owns and operates a large data center for its own uses, adding cloud computing offering on top of this infrastructure can provide new revenue streams at low incremental cost. It can help a company make the most out of their existing infrastructure.

3. **Defend a franchise**. A company with a well established conventional franchise should definitely consider creating a cloud based option of it. The fran-

chise could be used to migrate existing customers to the company's cloud environment.

4. **Attack an incumbent**. Moving into the cloud computing space would be a good move now, before a few massive cloud providers take over the business and destroy the competition. As cloud computing is based on the massive scales, giving a head start to the competitors could give them a large competitive edge that could be hard to catch. Offering alternative ways to utilize the cloud could be used to stand out from the competition.

5. **Leverage customer relationships**. Companies with extensive customer relationships should consider creating a branded cloud computing offering. Taking advantage of these long term relationships and trust, it could be easier to convince customers to migrated to the company's cloud environment.

6. **Become a platform**. A company could turn their cloud into a development platform (PaaS), offering customers the ability to build and host their applications in the cloud. Quick, easy and cost-effective (pay for what you use) development and deployment in the cloud would be the main selling points, offering a complete package.

3.1.1 Technical solutions

While cloud computing creates a new business opportunity for the infrastructure providers, it also moves the risks, problems and costs of acquiring, operating and managing the hardware to them. Cloud computing doesn't remove any of these problems, it just moves them from the service providers and users into the hands of the infrastructure providers. Though as we have concluded before, the infrastructure providers are specialized in this area, and are able to harness the economies of scale for the costs of the hardware, software, electricity, networking and operations [2], [9], [18]. They are also able to use virtualization techniques and statistical multiplexing to increase utilization rates beyond those of ordinary data centers [2], [16], [29]. Increased utilization rates bring down the costs of power, cooling and floor space, which is crucial in making cloud computing profitable [16]. These cost savings truly manifest themselves in clouds as they deal in massive numbers of hardware and supporting infrastructure.

The infrastructures are built on massive number of cheap commodity hardware, which creates a network that is spread thin and wide, and able to recover from hardware and connection issues [2], [34]. Virtualization is used abstract the servers,

storage devices and other hardware into a pool of resources that can be allocated on demand. This encapsulation of physical resources solves several core challenges for the data center managers and delivers advantages, such as: Higher utilization rates, resource consolidation, lower power usage/costs, space savings, disaster recovery/business continuity and reduced operations costs [28].

The cloud infrastructure needs to be geared for high levels of efficiency, service-level availability, scalability, manageability, security and other systemic qualities [28], [33]. The cloud needs to be highly automated and monitored to handle all the resource allocation, load balancing and various other jobs [29]. The high automation allows the cloud to be highly elastic (rapidly scaling up or down as needed), and makes it manageable [33].

Security is a top priority for clouds to convince enterprises and users to store sensitive data in the cloud [9], [29], [33]. Since the clouds are multi-tenant (the infrastructure and resources are shared among various customers), they must employ proper data access policies and protection mechanisms to create a secure multi-tenant environment. Each tenant could have their own protection requirements and trust relations with the provider, further complicating things. To provider security in the multi-tenant environment, Salesforce.com employs a query rewriter at the database level, and Amazon uses hypervisors at the hardware level to isolate multiple clients data from each other [29]. Virtual machine level vulnerabilities can be mitigated by deploying IDS/IPS (Instruction Detection System / Instruction Prevention System), by using secure remote access technologies like IPSec and SSL VPN and by implementing virtual firewall appliances in the cloud [31]. Virtual machines need strong isolation, mediated sharing and secure communication between other virtual machines. Even some of the clients' software running in the cloud could be malicious attackers in disguise [29], and hackers could be using the cloud infrastructure to organize and launch botnet attacks [6]. Overall the multi-tenancy model and pooled resources introduce new security challenges that require novel techniques to combat [6].

Grobauer et al. [15] note some vulnerabilities in the core cloud computing technologies that are intrinsic to them, or at least still prevalent in the state-of-the-art implementations. These include *escaping from a virtualized environment* (it's part of virtualization's very nature), *session riding / hijacking* (due to web applications being stateful) and *insecure or obsolete cryptography* (cryptography being absolutely essential for clouds to protect the confidentiality and integrity of all the customer's data). They further list vulnerabilities related to the essential cloud characteristics, such as *unauthorized access to management interface* (probability for this is higher than in tra-

ditional systems), *Internet protocol vulnerabilities* (cloud services are accessed using standard network protocols), *data recovery vulnerability* (reallocating resources from one user to another might leave previous data recoverable in the storage or memory) and *metering and billing evasion* (manipulation of the metering and billing data) [15]. Another issue is that there are no standardized cloud-specific security metrics for cloud customers to use, making security assessment, audit and accountability harder, even impossible [15].

Concerning data, the cloud provider must make sure that the customer's data is stored and processed in specific jurisdictions and that they obey local privacy requirements. Each customer's data must be fully segregated from other customer's data, and only privileged users should have access to it. Efficient replication and recovery mechanisms need to be in place to restore data in case of disasters and hardware failures. The data should be safe even if the cloud provider runs into problems (financial, etc) or is acquired by another company. If it is important for a customer to have investigative support of the cloud services, such support should be available and ensured with a contractual commitment [9].

Keller et al. [18] highlight the security issues of the virtualization layer. A successful attack on it could give the malicious party access to the memory of the system, compromising the confidentiality and integrity of the software and data (including encryption keys and customer data) of any of the virtual machines. They point out that many vulnerabilities have been shown to exist at the virtualization layer. They propose removing the virtualization layer altogether, and showcase their NoHype architecture as a replacement for it. The NoHype architecture aims to deal with the security issues of virtualization by resource isolation (e.g. only one virtual machine per processor core). They do note that their solution comes with some costs, like limiting the granularity of the core (e.g. can't sell 1/8th of a core) and over-subscribing a physical server (i.e. selling more resources than are available).

Authentication and identity management, access control, compliance to regulations, trust management and privacy requirements are among the security challenges that cloud providers must be able to resolve [15], [29], [31]. Particularly authentication issues are seen as one of the primary vulnerabilities of clouds in the cloud provider's experience [15]. Cloud security research should put special focus on getting tried-and-true security controls to work in a cloud setting [15]. Some of the inherent features of clouds provide security: Since the clouds are built to be loosely coupled, they are able to keep running and are put at less risk when one part of the cloud goes down or gets targeted by malicious attackers. The abstraction and virtualization that clouds are built upon avoids exposing the details of the

underlying implementations and offers security by isolation [14], [33].

The data centers are using modular approaches for provisioning of the hardware resources. Example of this are Points of delivery (PODs) that encapsulate servers, storage, networking and the management of these resources. These environments can be optimized for specific workloads (e.g. HTTP or HPC) or specific capacities (e.g. a number of users or transactions). Applications in these environments can scale independently, and additional PODs can be added if more resources are needed. This provides both availability and scalability for the cloud [28].

Special techniques are being used to deploy software on the virtual resource pools. Software components, data, server and storage pools and other cloud resources are being combined into **software packages**. These packages act as a software delivery mechanism that simplifies and accelerates the installation of everything from operating systems to applications and end-user data. They make efficient resource allocation, re-use and management possible [28]. Similarly **machine images** are being used to deploy application development payloads in the cloud. These machine images can contain user-specific applications, libraries, data and associated configuration settings. Well-known examples of these are the Xen images, and also the Amazon Machine Images (AMIs). The AMIs are built around a variety of kernels, and you can choose from public preconfigured images or modify one for your own needs [28].

Cloud computing can bring some changes to the hardware and software of the data center infrastructures. Hardware systems should be designed at the scale of a container. The focus should be at the horizontal scalability of multiple virtual machines over the efficiency of a single VM. Energy efficiency is important, idle portions of the memory, disk, network and other hardware need to be put into low power mode. The software needs to be aware that it is no longer running on bare metal, but on virtual machines. The infrastructure software needs to have a built in billing mechanism to capture the required information for the pay-per-use model [2], [6]. Getting all this information from the virtualized environment of the cloud can be a lot more complicated than for ordinary data centers that base their cost on consumption of static computing [6].

3.1.2 Classes of utility computing

One interesting thing to note of the different cloud offerings is what Armbrust et al. [2] call ”classes of utility computing” (Dong Xu [36] calls it ”classes of cloud platforms”). The cloud infrastructure, platform or service can vary on the level of

abstraction presented to the programmer and the level of management of resources [2]. Armbrust et al. and Dong Xu use the Amazon EC2, Google App Engine and Microsoft Azure as examples to highlight these differences.

The Amazon EC2 is the most flexible platform, allowing the user to go with the kernel and software stack of his choice, allowing it to run any kind of applications with very few limits [2], [36]. On the other hand, this freedom makes it difficult for the platform to offer automatic scalability and failover, since much of the semantics of replication and state management are application-dependent [2].

The Google App Engine is at the other end of the spectrum. Built to run traditional web applications, it enforces several constraints on the applications that can be developed for it. The application must follow a stateless computation tier and a stateful storage tier separation, and thus isn't suitable for general-purpose computing. However, due to these tighter requirements the App Engine is able to achieve impressive automatic scaling and have a high-availability mechanism [2], [9].

Microsoft Azure is somewhere between these two. Azure applications can be written in .Net, Java and PHP (among other languages), and they are compiled to the Common Language Runtime, a language independent managed environment. This allows the system to support general-purpose computing. The user has no control over the underlying operating system or runtime. The system offers some automatic network configuration, failover and scalability, but requires the developer to declaratively specify some application properties to do so. This puts Azure at somewhere between the flexibility offered by the Amazon EC2, and the programmer convenience offered by the Google App Engine [2], [36].

Armbrust et al [2] argue that none of these "utility computing classes" stands over the others. Just like in the traditional programming, both low-level and high-level (and anything in between) languages have tasks where they are best suited. If performance is a top priority, a low-level language will offer the best results, while a web-application or graphical user interface are best achieved with higher level languages. Similarly, different applications and tasks in the cloud will require different classes of utility computing [2]. Sun Microsystem's Cloud Computing primer [28] suspects that over time the level of abstraction that the developer interfaces with to move gradually upward, as more and more clouds (especially private clouds) will offer higher-level development environments. Armbrust et al. [2] believe that software management costs (upgrades, applying patches, etc) will be lower for the high abstraction level managed environments like Google App Engine, though quantifying these benefits in a clear way might prove to be difficult [2].

3.1.3 Service-level agreements

A number of the definitions of cloud computing we went over earlier mentioned Service-Level Agreements (SLAs) as an important feature of cloud computing. Since unpredictability is a fact in distributed computing environments like clouds [26], [28], the SLAs are in a very important role in assuring a certain Quality of Service (QoS) level in the cloud. A guaranteed QoS and SLA enforcement are essential for clouds to gain the confidence of the cloud users and of people who are still thinking about making the move to the cloud [6], [7], [33], [36]. Typical QoS goals can be ”storage should be at least 1000 GB”, ”bandwidth should be at least 10 Mbit/s” and ”response time should be less than 2 s” [22]. Essentially the SLA is a service contract between the cloud provider and the customer. It defines the level of service that the cloud provider has promised to deliver to the customer. It deals with the services, priorities, responsibilities, obligations, guarantees, warranties and service pricing [6], [7], [29]. The SLA also states what sort of compensation the customer gets should the cloud provider violate these terms and not be able to hold up to what was promised.

The unpredictability of the cloud can present itself in availability and / or performance. Cloud operators often base their SLAs on availability and uptime of the service, ignoring performance and throughput [26], [33]. This could be discouraging for a number of cloud users, including researchers whose experiments require guaranteed performance to deliver repeatable results. Even more common cases like games and web-applications could want assurances on how fast the content is being served [26]. Other aspects like security, privacy and trust are inherently non-quantitative and can be difficult to bargain [29], [31], and could lead to disagreements whether the SLA has been violated or not. The SLAs need to be defined in a way that has balance between expressiveness and complicatedness, so they can be enforced by the resource allocation mechanism of the cloud, yet remain relatively simple to be weighted, verified and evaluated by the customer [6].

Having the SLA written down wouldn’t mean much if there wasn’t a reliable way of measuring the service level and automatically detecting SLA violations. This has sparked research into autonomic SLA violation detection in cloud infrastructures [7], but also into adaptive resource configuration in cloud infrastructures to avoid such violations before they can happen [22]. There can even be trust issues regarding the SLA monitoring – the customer might not completely trust a measurement system managed solely by the cloud provider. This might require for a third-party mediator to handle the measurement and report violations [29].

The monitoring of the SLAs creates some challenges, like how to map low-level resource metrics to the high-level SLAs, and finding the appropriate intervals for the measurements so they can detect SLA violations early, but not become too intrusive to the system [7]. It might not be obvious how the low-level parameters like bandwidth, storage, CPU power and memory exactly affect the high-level parameters like response time. E.g. Does ”response time < 2 s” translate into ”memory > 512 MB” and ”CPU power > 8000 MIPS” or rather ”memory > 4096 MB” and ”CPU power > 1000 MIPS”? These translations should be managed by an autonomic system with little human interaction to guarantee scalability and strengthen the dynamic behavior and adaptation of the system [6], [22]. The SLA management needs to be both flexible and and reliable [7]. Advanced SLA mechanisms need to constantly incorporate user feedback and customization features into the SLA evaluation framework [6].

The SLAs are not only used to provide guarantees to cloud users, but they are also used by the cloud providers themselves to efficiently manage their infrastructure in an environment of competing priorities like energy efficiency, the attainment of SLA agreements and the delivery of sufficient elasticity. They also have an important role in more complex cloud environments like a federated cloud where the workloads are being in- and outsourced between clouds [7].

3.2 Service providers

The service providers [33], software as a service providers or cloud users [2], or the median users [14]. All of these refer to the companies that use clouds as a means to deliver their services to the end users. Much of the hype surrounding cloud computing has focused on the benefits that the service providers can potentially gain by outsourcing all or parts of their own hardware / infrastructure into the clouds. Not just mere promises, cloud computing can provide many benefits from small firms to large enterprises, and many of these changes can seem quite radical. One could possibly compare it to the hype around the dot-com bubble of the late 90’s and early 2000’s, though the final outcome of the cloud boom remains yet to be seen.

3.2.1 Benefits of cloud computing

The first and most obvious advantage for a company to use clouds is getting rid of all the capital expenditure. No hardware acquisition or maintenance costs, no

software licences or upgrades to manage (in the PaaS offering), no employees or consultants to hire (for the infrastructure), no facilities to lease and no power bills [2], [16], [28], [29], [31], [34], [38]. Instead you pay only for the resources you use, only when you actually use them. In essence it fills the utility computing model: Instead of any fixed costs, you are paying for computing resources as a metered service, much like you would pay for electricity. This eliminates any hidden costs, or unpredictable extra costs that could result from things like hardware failures. It can make the launch of a service much easier and predictable, since provisioning doesn't need to be planned far ahead [2]. Getting rid of all the upfront costs is especially attractive to startups and smaller companies, who would have to invest a lot of money to even get their service started – without any guarantees the service would ever properly take off. On the other hand, it eliminates the scenario where the service becomes wildly popular and exceeds all expectations, and the infrastructure wouldn't be able to handle the load, resulting in lost customers and revenue [2].Thus not only does it save on the upfront costs and make planning things easier, but it can reduce the risks involved in launching a new service. Or on the other hand, it can encourage companies to take more risks.

The other much hyped feature is the "instant and infinite" automatic scaling of the cloud, or on-demand scalability [2], [16], [26], [28], [34]. When there is a sudden and unexpected spike in the number of users of the service, the cloud can automatically and rapidly scale the service up to match the needed extra resources. When the peak is gone, the service automatically scales down again. The "instant and infinite" scalability of course depends on the cloud having enough free resources to take on more work at that time. Even the clouds can run out of resources, and might need to outsource some of the work to other clouds (federated cloud), or possibly be forced to pay compensation for violating the service-level agreement. This elasticity is possibly the biggest advantage that a company could gain from using a cloud, since ordinary data centers are estimated to reach server utilization rates from 5% to 20%, leaving expensive equipment idle a lot of the time. These low numbers can be explained by the observations that for many services the peak workload exceeds the average workload by factors of 2 to 10 [2], [34].

To increase the utilization rate a company would have to deliberately provision for less than the expected usage peak, which would result in poor service during peak times, possibly causing affected users to abandon the service. On the other hand provisioning for the usage peak (or even over-provisioning for it), which is much more common than under-provisioning, will cause the utilization rates to go down. So no matter how the provisioning is done, the data center can never reach

very high utilization rates and at the same time keep the service availability high. Not to mention predicting the usage peaks can be an impossible task in itself [2], [4]. Since the cloud automatically scales to match the needed resources, not more and not less, it can actually provide a 100% utilization rate, at least from the service provider's perspective. The cloud provider might charge a premium for assuming all the risks related to provisioning and utilization rates [2].

One more extra layer of difficulty related to utilization is that applications rarely use different computing resources (CPU, memory, network, etc) equally. Designing a system that kept the utilization of all these resources high would be very difficult. A cloud can charge each of these resources separately based on their actual usage, again reducing costs and the complexity of planning the infrastructure [2], [10]. The decision whether to use clouds or a private data center wouldn't have to exclude each other out, the company could retain their own data center and outsource work to clouds when it seems that the data center can no longer handle it (the hybrid cloud concept, or data center "augmentation" [28]). This would provide on-demand scalability and yet allow the company to retain control over their own infrastructure, though it would introduce its own set of problems like design and synchronization issues [27], [28].

Using clouds can greatly speed up application development, testing, deployment and shorten the time to market [16], [26], [28]. This can allow companies to seize new business opportunities quickly and effectively. It allows companies to act in a much more dynamic way, introducing new services and scaling down old ones according to their needs. All of this can be done very quickly and easily, with no additional costs that would result from having to manage the hardware and software yourself. This business agility can create new business opportunities to build new and better network services in less time and for less money [2], [28]. Computing resources can be acquired for short-term use (e.g. for large scale batch-processing and analytics jobs), and released when no longer needed [2], [28].

Having the software, services and data in the cloud allows companies to ignore things like taking care of backups and dealing with security issues like viruses [34], [38]. The complexity of managing hardware and software by the company itself is gone [14]. All of these responsibilities can be shifted to the cloud providers. Selling or licensing software as a product to the end user requires it to cope a lot of different operating environments (different hardware, different operating system, different drivers, etc). Deploying the software into the cloud and offering it as an Internet based service suddenly removes most of these worries. The software can be developed, tested and run on a unified platform tailored to it's needs. Updat-

ing the software can also be done in the cloud, saving this trouble from the clients, and making sure everyone is running up-to-date version of the software [2], [17]. Though it wouldn't remove the problem entirely, the server-side software would still need to be able to interact with a variety of clients, like different web-browsers, and some users could be using older versions of those web-browsers [17]. Though it would certainly lessen the problem of having to develop and test software for a large variety of platforms, at least in the PC world. It is good to note that the aspect of offering an application as an Internet based service is not something entirely new and exclusive to clouds, as the concept of SaaS has been around long before clouds. However clouds are new in that they enable more companies without their own infrastructure to deploy applications as SaaS [2].

Due to cloud operators being able to harness the economies of scale [2], [9], [18] clouds can be offered at a price that is very competitive to having your own infrastructure. One could say that switching over to a cloud reduces costs of operation [20], [27], [28], [29], [36], [38] and not be too far off from the truth, at least at the current state of things. Armbrust et al. [2] present the following equation (Figure 3.2) to calculate which solution (cloud or data center) will yield higher profits for the service. The equation assumes revenue is proportional to user hours and that the service will be under varying demand over time (thus utilization rate becomes a significant factor for the data center side). "Utilization" in the equation is the average utilization rate, including nonpeak workloads. The side with the greater value should yield higher profits [2].

$$\text{UserHours}_{\text{cloud}} \times (\text{revenue} - \text{Cost}_{\text{cloud}}) \geq \text{UserHours}_{\text{datacenter}} \times (\text{revenue} - \frac{\text{Cost}_{\text{datacenter}}}{\text{Utilization}})$$

Figure 3.2: Cloud vs. data center equation [2].

Dillon et al. [6] note that a company considering to move to a cloud must consider the trade-offs between computation, communication and integration. For example using the cloud can significantly reduce the infrastructure cost, but the cost of data communication (transferring data in and out of the cloud) can be higher. Using clouds works better for applications that are CPU-intensive, rather than for applications that are data-intensive [2], [6]. As the role of data and data management will continue to grow, exploiting data locality is important [2], [6], [9]. Since clouds often use proprietary protocols and interfaces [6], [20], [28], [31], [32], [36], the costs of data integration can be substantial, as developers have to work with various cloud-specific APIs and develop ad-hoc adaptors [6].

For long term use, using clouds enables fluid fine-grained economic models, and the elasticity can transfer the risks involved with provisioning and utilization away from the company. Even if you were able to buy and set up servers cheaper than using the cloud for the same period of time, the transference of these risks can more than make up for it [2]. Using clouds can also track the changes of hardware prices more closely, potentially passing savings from declining hardware prices more effectively to the customer [2]. Though this could also work in reverse, if certain component prices like memory suddenly experienced a high rise in prices? Since the clouds operate on massive scales, surely a stark rise in component prices would be reflected to the customers. New and more efficient hardware and software solutions becoming available for the cloud providers can pass some of the savings gained from them to the cloud customers immediately and without any capital expenses [2]. Overall the cost of operation (for better or for worse) is more likely to follow the current trends.

Cloud computing can make software development faster and easier, and provide a wider selection of lightweight and agile development tools. This can enable new programming and business models [28]. For example the Google App Engine and Salesforce.com offer online development environments and tools to easily and quickly develop and deploy applications in the cloud. As the cloud adopts new technologies, so will new infrastructure services, options, development tools and other features become available for the cloud customers immediately and with no capital expenses [2].

3.2.2 Challenges and concerns

While the benefits of moving applications and services to the Cloud can be significant, there are technical, security and even legal issues that can come with it.

Guha et al. [16] explore the impact of Cloud Computing on the software engineering process. As most software project failures are caused by the lack of communication and coordination between all involved parties [16], Guha et al. stress the importance of including the cloud provider in all the different phases of software development. Since the cloud provider knows the details of the architecture, they can provide key information for the requirements gathering, planning and design phases of software development. Among many things, they can provide information regarding: 1) developer requirements, 2) component reuse, 3) cost estimation, 4) schedule estimation, 5) risk management, 6) configuration management, 7) change management and 8) quality assurance [16]. Since the infrastructure is in the hands

of the cloud provider, they should be included in the deployment and maintenance phases. Issues like data protection, security, availability and reliability of the resources need to be managed with the cloud provider. These things are usually covered in the service-level agreements as mentioned earlier.

The amount of interaction between the software engineers and the cloud provider naturally depends on the type of the cloud, with a public cloud requiring the most interaction (resulting in a more complex software development), while a private cloud requires the least interaction [16]. According to Guha et al. [16], the cheaper computing costs offered by a public cloud will always outweigh the extra complexity of the software development process it introduces. On the other hand, the cloud can also have benefits for the software development process. Since the software is easily accessible for everyone, coding, testing and validation can be done more cost and time effectively [16]. It is also likely that as more and more software is developed for clouds, the whole software development process for it will mature and be refined to meet these new challenges.

In their paper [16], Guha et al. delve further into this issue, and propose a model that extends the Extreme Programming (XP) agile model for cloud platform development. Their model includes the cloud provider in all the development phases, and details their roles in them. This role separation can be seen in Table 3.1 [16].

Activity	**Roles**	
	SW Developer	**Cloud Provider**
Requirement Gathering	Elicitation	Resource Accounting Virtual Machine
Analysis	SW Modules	SW/HW Architecture
Design	Interface Design Data Types Cost Estimation Schedule Estimation	Component Reuse
Construction	Coding Integration of Web Services	Implementation Details
Testing	Unit Test Integration Test	Integration Test
Deployment		Operation & Maintenance

Table 3.1: SW Engineering- Role Separation [16].

Guha et al. [16] estimate that the extra effort that results from the interaction with the cloud provider and the lack of documentation of implementation details of the

web services will increase the complexity of the project. On the other hand, they estimate that component reuse of web services will significantly reduce the amount of code that needs to be written from scratch. Overall the work required to develop the software will reduce, but a new level of communication and coordination with the cloud provider will be introduced [16]. The picture might get even more confusing if more than one cloud provider is involved in the same project: A client subscribes to an IaaS from one provider, couples it with a PaaS from another provider, and mixes in various pieces of SaaS from a third provider [29]. This could raise various security and integration problems for the project, and the required communication and coordination would increase dramatically.

One challenge that many enterprises moving to clouds will face is having to migrate data from their legacy applications into the cloud. The complexity and cost of this effort can be significant and must be taken into consideration in the decision to move into the cloud. Many companies have started to specialize in data integration for clouds to capitalize on this new business opportunity [2]. Moving locally run applications to the cloud could also force companies to master new languages and operating environments [17], though many of these techniques have already been used for developing web applications for years. Also, clouds are often designed to make software development and deploying easier and faster, as opposed to making the process harder [28], [33]. Though this can depend on the abstraction level of the development environment. The integrated, optimized and open-source AMP (Apache, MySQL, PHP/Perl/Python) stack that is the most popular platform for developing web applications and services today might be replaced with new and even more lightweight and agile tools [28].

The one big concern about cloud computing are the security and privacy risks that the unique architectural features of clouds raise [6], [9], [17], [20], [29], [31], [37], [38]. Since all the data, applications and services from various different customers are centralized and run on the same physical hardware, there are of course concerns about privacy and security. Do only the authorized entities have access to our data? Since we aren't in charge of the infrastructure, can we trust that the cloud operator has employed appropriate mechanisms to enforce strict rules on data accesses and security? Are the customers even allowed to know any of the inner workings of the cloud, or do they simply have to trust the cloud provider? Putting very sensitive data out there somewhere in the cloud certainly requires the customer to have a lot of trust in the cloud provider's technical competence and economic stability [29]. Dillon et al. [6] note how a survey conducted in 2008 revealed that organizations still had security and privacy concerns about moving their data to clouds. Marginal

functions were being outsourced to the cloud, while core activities were kept in-house. Though the survey also showed that in a few years time a growing number of organizations (31.5%) planned to move their storage capacity to clouds, but this number was still lower than for peripheral and other less important functions [6]. The service-level agreements can cover some of these security and privacy concerns and help put the customers worries to rest.

Though a service provider has to ask itself whether their own private data center or the cloud is able to provide better security. After all the cloud are massive in scale and built by companies who have specialized in security due it being absolutely essential to their business. A cloud provider could not afford to suffer massive security breaches [31]. Cloud providers are most likely able to provide better physical security by having surveillance cameras and extra security personnel on-site. They also employ more advanced network security measures than simple firewalls. All of these are cost prohibitive for a single organization, but in a cloud the costs are spread across all the customers, making them insignificant [18]. On the other hand, a massive cloud facility might be a lot more likely to be targeted by capable malicious attackers as the ”rewards” for a successful attack would be far greater. In the end, which infrastructure would be able to provide better security and data protection might not be completely obvious [20].

Though the security and privacy are not only the responsibilities of the cloud provider. In PaaS and especially in IaaS the customers themselves are responsible for designing and building the applications to be secure. The cloud provider is responsible for isolating each customer's applications and data from each other, and they must provide some basic, low-level data protection capabilities so unauthorized access to the data in the cloud cannot be gained from outside the applications themselves [29]. This doesn't change the fact that if the customer built a poorly designed application with poor security, the responsibility of this is not the cloud providers. Their responsibility is to make sure a poorly designed application running in the cloud does not compromise other applications, services and data in the cloud.

There are some issues that might not come to mind at first. What if the cloud provider goes bankrupt, or you just want to move your applications and data to another cloud provider? Can you get your data, services and applications out from it? How much of them can even be transferred to another cloud? Since some high level PaaS offerings like the Google App Engine put limitations on how the software must be designed [2], would it even be possible to run applications designed for it anywhere else? How much would they need to be reworked to be used elsewhere?

Obviously the most low level IaaS offerings like Amazon EC2 would put the least limitation on the applications and services, which would make moving them to another cloud easier.

Ideally a variety of cloud providers would offer standardized open-source interfaces to common services. The user would be able to choose where to put their applications, easily move them from one cloud provider to another if necessary, and combine services from different clouds. Though this might be wishful thinking, as most clouds are proprietary and the underlying services like storage and databases can result in significant lock-in to the selected cloud [6], [20], [28], [31], [32], [36]. The proprietary and differing cloud APIs also create challenges for organizations trying to integrate cloud services with their own local legacy systems or with other cloud services [6].

There has been some work done towards more open-source cloud interaction. Eucalyptus is an open-source software for creating on-premise private and hybrid clouds. It supports the cloud interface of Amazon Web Services (AWS), works with different types of hypervisors and virtualization technologies and allows the on-premise clouds to interact with public clouds through a common programming interface [36]. Perhaps an initiative like this could in future grow to support a wider range of clouds. Other solutions like a virtual infrastructure (OpenNebula), a higher level abstraction layer, the Unified Cloud Interface (UCI) and Sun Open Cloud API have been researched and developed to hide the heterogeneity of the different proprietary clouds and allow the development of cloud-provider independent applications [6]. Dillon et al. [6] note that most standardization efforts have been on the IaaS side, with SaaS receiving less focus. They find that PaaS interoperability in particular might have big problems ahead since the entire software development lifecycle is managed in the cloud, which would require reaching uniformity in all the aspects of software development and deployment.

Cloud computing becoming more prominent could present a future challenge for the open-source community. As the focus is shifting away from things being run locally, open-source applications might need to figure out a way to compete with cloud applications like Google Docs and Microsoft Office 365. It's one thing to create non-profit applications by dedicated volunteers, but it wouldn't be possible to have a non-profit cloud, at least not one large enough to stand against the massive commercial clouds [17]. Maybe donations could cover the costs of hosting the service or application in some commercial cloud, or donating old hardware or money for some sort of non-profit cloud? Possibly some sort of peer-to-peer (P2P) cloud solutions where volunteering computers can share idle resources to form a cloud?

Fouquet et al. [10] explore the idea of an open-source video streaming application that integrates cloud-based infrastructure with P2P systems. The system envisions of IaaS being made available directly to end-users, without the involvement of commercial service providers in the middle. As an example they use, a group of users are connected to a video stream. Clients A and B receive the steam from the original sender, who has bandwidth to support two clients. Client C receives the data from client B. As client D connects to the stream, the upstream bandwidth becomes insufficient. The P2P applications notices this and asks the clients if any of them are willing to sponsor a relay server, making the video quality better and smoother for all the viewers. If any of the users decides to pay the required small fee (i.e. 1 dollar per hour), a suitable virtual machine running this relay server is spawned onto any of the clouds supported by the application. This way the end-user is directly purchasing computing resources from a cloud, through the application. Spawning cloud servers to act as super peers in the P2P systems can give the networks urgently needed resources [10].

Users could be motivated to pay the small fee to launch relay servers by giving them highest priority for the service and a good position in the distribution tree, guaranteeing a good video quality. They would also not be dropped out of the stream when resource are running out. The freeriders would be receiving a lower quality video and be the first to cut out if they refuse to pay the fee when resource are running out. This way the application would be using a distributed voluntary payment model, where users can cooperatively deliver services that would otherwise require a large infrastructure [10].

There are some issues that would need to be worked on for this sort of system. For one the cloud interface is not designed for average users, but rather for developers. It would need to be simplified. There would also need to be a way to reliably identify the user and obtain his permission for launching the virtual machine, as malicious parties might be trying to initiate botnet style activities at other users' expense. Some users could also try to insert malicious relay servers into the video distribution tree, or users could try to cheat the application to gain unfair advantages in the system (e.g. higher priority, better video quality) [10].

Though the threat to open-source might seem pretty far fetched today, in the distant future it could become a reality if cloud computing really starts shifting the focus of the hardware and software industry away from personal computers and towards the clouds. Open-source activist Richard Stallman criticises this movement by noting that you lose control over to the provider of the service or server that you use, and that you will become dependent of them and that you are at their mercy [2],

[10]. Fouquet et al. [10] note that open-source software and SaaS are fundamentally incompatible, due to the principle of open-source software being free. SaaS cannot be offered free, as building and maintaining the infrastructure to support it has an operating cost. Even if the service was offered for free for end users, in the end **someone** has to pay the expenses to keep it running.

3.3 Service users

When we are looking at the benefits and concerns that cloud computing brings about to the service users, or the end users, we need to realise that we are mostly talking about the SaaS model [2]. SaaS as such is nothing new, web applications and services have been around long before clouds. From the eyes of the end user the impact of cloud computing might seem pretty insignificant compared to web applications hosted in ordinary data centers. Regardless, SaaS is an integral part of cloud computing. Clouds will likely be a driving force behind making more and more applications be delivered as web services, as they allow companies without an infrastructure of their own to deliver applications as SaaS [2].

One of the biggest advantages of SaaS for the end user are mobility and accessibility – the service can be accessed anywhere on any device with a web-browser (assuming you have Internet connection available there). It can make sharing data and collaboration much easier [2], [17], [20], [34], [38]. Take for example multiple people editing the same document in Google Docs. No longer do each person have different versions of the same document, edited at different times, and trying to keep that file up to date by sending these version to each other through email. All changes to the document are done centrally in one location, and everyone can see the changes as soon as they happen. The files are also available wherever you go, without having to store them on an USB stick and carry it around. Cloud storages can provide large amounts of space (seemingly endless [20]), and provide a dependable and secure place for backups [2], [38].

Though all the positive qualities of mobility and accessibility can turn on their head if the user is in a region with bad Internet connection, or in a situation with no Internet connection available at all. If all the applications and data were in the cloud, a situation like this would completely prevent access to them. Heavy dependence on clouds can turn Internet connection comparable to electricity. Without it you wouldn't be able to access any of your services or data in the cloud. Imagine a critical monitoring application that has to be available and running at all times. In case of electricity going out, there would be a backup generator to keep the system

functioning. If such an application was running in the cloud, what would be the backup solution if the network connection was lost? This is a simplified example, as such an application would probably not be run in a cloud. But it does raise the question of what applications are safe to be run in clouds, and would they need fail-safe measures if connection to the cloud was lost.

SaaS can also liberate end users from having to manage the software. Updating is done automatically and the software is always kept up to date. If cloud computing was taken to the next level, where most of the computations are done in the cloud and the user just has a "dumb" input output device [34], [37], [38], even more responsibilities would be shifted away from the end users. No longer would you have to worry about purchasing, installing, configuring and updating hardware or software, or deal with issues like viruses and spyware [17], [20]. Though this aspect might be less than exciting for hardcore users and hobbyists who are capable of doing these things themselves, and want to be in control of their own computers. For them cloud computing would mean a real loss of control [20]. A kind of similar thing has happened with modern cars. Car enthusiasts used to be able to fix any problems and tune their cars themselves. Modern cars are full of electronic systems that only the certified people can fix, leaving even an experienced car hobbyist scratching his head when the car stops working.

Cloud computing could be combined with locally run applications as well. The application would be installed and run on a local device, but one or more of its features could draw upon cloud computing resources. Tasks with a high computations-to-bytes ratio like 3D image rendering or massive batch jobs that would take days to finish would be good candidates for features that could be outsourced to clouds [2]. Though creating a payment model for this sort of application would require some thought, since some customers could be using the cloud features a lot while others might never use them. With a subscription based payment model it would be easier to spread the cost across all customers, but if the software is sold for a one time fixed cost it would be harder. Perhaps a separate small cost could be attached to using these cloud based functions. The user could be given the option to do the operation locally (possibly taking days to finish), or pay a small fee to have it done in the cloud. Fouquet et al. [10] explored this sort of idea in their P2P video streaming service that we covered earlier. In this way cloud computing could be used to enhance local applications instead of completely changing them to web services.

Having someone else be in control of all your data can of course raise many questions about privacy and security [6], [9], [17], [20], [29], [31], [37], [38]. Can you be absolutely sure your data is visible only to those who you want to be able to see

it? How can you know sensitive information isn't being misused? Are you even informed if your documents have been given to someone? Even if you hadn't put information in the cloud yourself, someone else might have (e.g. bank storing your personal financial information or hospital storing your medical records) [20]. Can you trust security is properly taken care of in the cloud? With clouds in control of all your hardware and software you would be putting a lot of trust in their technical competence and economic stability [29]. What if the cloud decides to change their terms of service? If you refuse to accept the new terms, can you get your software (or software licence) and data out? Where would you move them? Most clouds are proprietary and the underlying services like storage and databases are designed in a way that can result in significant lock-in to the selected cloud [6], [20], [28], [31], [32], [36]. What if you failed to pay your bill, would you lose access to your documents? Cloud computing can raise these questions about control and ownership [17].

SaaS runs and stores the applications and data remotely on a server, but the question is whether that data would be safer located in a cloud or a private data center? One could argue that a cloud provider specializing in building massive clouds and hosting software from many different customers has to have made security their top priority. A cloud provider could not afford to suffer security breaches that would put their customers data at risk [31]. Cloud providers are most likely able to employ better physical and network security measures than a private data center [18]. On the other hand, a global massive cloud hosting thousands of customer's software and data might be more likely to be targeted by malicious attacks than a smaller, local data center. Which infrastructure would be able to offer better security might not be completely obvious [20]. Availability and reliability might be better in a cloud than a private data center, since a private data center would be much more likely to run out of resources (assuming it wasn't outsourcing the extra work to clouds). Hardware failures in a massive cloud would barely be noticeable since there would be many servers ready to take their place. In a private data center the effects of hardware failures would most likely be felt harder.

One new thing clouds specifically have enabled on the SaaS front are resource intensive applications appearing as services. Cloud-based gaming services like Gaikai and OnLive are probably the most radical examples of this, running modern games in the cloud and streaming the gameplay as video for the clients. Though the potential hasn't been fully realised yet, they do prove the concept is doable. OnLive did suffer quite a setback recently, running out of money and laying off all of their staff. The company ended up being sold to a venture capital group for 4.8 million US dollars, when analysts had previously estimated it's value to be 1.8 billion US dollars

[3]. The reasons for OnLive's failure were trying to be available on too many devices (PC, Mac, smart phones, tablets, TV and even their own console), poor offering of current games and technical issues related to the cloud like lag and games being run on lower resolutions and visuals to make the streaming smoother [30]. Gaikai fared better and was sold to Sony for 380 million US dollars a few months earlier [3]. Gaikai will be part of the upcoming PlayStation 4 console, allowing demos to be streamed through it and being used for social gameplay elements [24].

One very significant element that cloud-based gaming (or cloud-based software in general) could bring about is pretty much the elimination of software piracy. Since the entire game is located and run in the cloud, and only the video is streamed for the client, getting such a game to run locally with no connection to the cloud would be quite an undertaking. The games could be designed to run on very specific environments, so even if the game was somehow leaked, it would take large amount of work to get it to run on ordinary PCs. This aspect could make cloud-based gaming very intriguing for game developers and publishers. It might also kill the used games market, another thing that game publishers have been trying to combat. For the gamers it could mean losing the ability to sell or loan games. It could also severely limit modding possibilities of games. Games would require a constant Internet connection to the cloud, something which would simply be a very bad thing for many developing countries. Even areas in developed countries like the US have unreliable and poor Internet connections. Recent launches of games that require constant Internet connection (Diablo 3 and SimCity) have also shown that provisioning for the massive number of clients in the first couple of weeks can be quite problematic [11]. For a cloud-based gaming platform the problem would be even bigger, as the bandwidth required to stream real-time video for millions of users would be very high. Though some of these aspects are already true for the popular Steam platform, and they haven't stopped it from becoming the leading digital distribution channel on the PC.

Everything being run in the cloud could also mean losing access to older games that the publisher no longer deemed worth keeping available in their service. On the other hand, a cloud service could be used to give users access older games and software that their modern computers or consoles would no longer be able to run. Often this sort of thing requires a lot of know-how and work from the user, possibly even older hardware. A cloud service dedicated to running older games and software could take away all of these problems from the user. Since one of the reasons for OnLive's failure was not being able to run modern games with high hardware requirement smoothly [30], wouldn't running older and a lot less resource intensive

games be a perfect fit for a gaming cloud service? In fact Sony is planning to use Gaikai to allow the PlayStation 4 console to run games from the older PlayStation consoles, without having to build hardware compatibility or emulation software into the console itself [12]. Smaller downloadable titles available in the different online services (e.g. Steam, Xbox Live, PlayStation Network) could also be a good fit for a gaming cloud service, removing all the hassle of downloads, allowing a quick and easy access to them.

4 Technologies and paradigms comparison

In this chapter we take a closer look at the technologies, methodologies, computing models and paradigms that are related in one way or another to cloud computing. We try to figure out how each of them relates to cloud computing: Are they an essential part of cloud computing? Do they share some characteristics with cloud computing? What are the differences between them and cloud computing? How do they present themselves in cloud computing? Since many of these terms appear in discussions about clouds, it is important to understand their fundamentals and how they relate to cloud computing. This chapter will hopefully clarify the relationship between these models and what they mean to each other. Some of the topics from this chapter were already covered to some extent in the previous chapter (Software as a Service (SaaS) and utility computing), but here we will look at each of them specifically. We will also take a look at a few models that have not yet been introduced in this paper: Grid computing, edge computing and Service-Oriented Architecture (SOA). Grid computing in particular is of great interest, as it can in some ways be seen as the precursor and even a rival to cloud computing, and it shares many characteristics with clouds.

4.1 Cloud computing and Software as a Service (SaaS)

The Gartner IT glossary [13] defines Software as a Service as follows:
"Software as a Service (SaaS) is software that is owned, delivered and managed remotely by one or more providers. The provider delivers software based on one set of common code and data definitions that is consumed in a one-to-many model by all contracted customers at anytime on a pay-for-use basis or as a subscription based on use metrics."

We already touched the SaaS concept and its role in cloud computing in the previous chapter. Looking at the division by Armbrust et al. [2] (see Figure 2.1), SaaS is the other half of what makes up cloud computing (the other being utility computing). Service providers deliver applications to end users over the Internet as services. Whether that service is hosted in an ordinary data center (not cloud computing) or in a cloud makes no difference, the concept is still SaaS. SaaS is fundamentally tied to cloud computing, since any application or service deployed in a cloud automatically qualifies as SaaS.

SaaS has been around long before clouds were even talked about, in the form of web applications and services. Cloud computing does not change the SaaS concept itself. Generally it makes no difference to the end user if the service is being run in a private data center or if it has been outsourced to a cloud. What clouds do change is making SaaS more prevalent, since companies without their own infrastructure can now deliver their applications as services, through clouds [2]. The benefits of clouds such as cost reductions and the automatic scalability of the infrastructure will surely promote more and more applications to appear as SaaS. Clouds can also allow more resource intensive applications (like games) to be offered as services, something that ordinary data centers would be hard pressed to handle.

In summary, SaaS is an integral part of cloud computing by the cloud's very nature. SaaS has been around before clouds, and cloud computing does not change the concept itself. Clouds broaden the type of applications that can be delivered as services and make SaaS more prevalent and attractive for service providers.

4.2 Cloud computing and utility computing

Utility computing is the other half of cloud computing in the definition by Armbrust et al. [2]. Infrastructure providers sell computing resources (CPU, memory, storage, bandwidth, etc) as a utility to the service providers (or in some cases to the end users, e.g. for batch jobs), and these resources are typically billed in a pay-per-use model. Just like SaaS allows *end users* to offload some of the problems into the hands of the *service providers*, utility computing allows the *service providers* to offload some of their problems into the hands of the *infrastructure providers* (aka *cloud providers* in cloud computing) [2]. This creates the three level service hierarchy as seen in Figure 2.1.

The utility computing concept was talked about all the way back in the early 60s [9], though these talks were mostly speculation and predictions. The 50 year old time-sharing service model where users with terminals communicated with a centralized site where all the computations were done also has connections to the utility computing model [17]. At this time personal computers hadn't arrived yet, so the predictions at the time about public utility computing were most likely based on this time-sharing service model. An evolved version of it, since access to the computations were limited to select group of people and institutions. The first computing model to really bring forward utility computing was grid computing in the mid 1990's. Though even the grid concept started out as a cooperative effort by organizations and institutions (usually academic or government) to help each other out by

sharing idle resources [9], rather than to form a publicly available generic utility for mass consumption (though commercial oriented grids have appeared since then).

Cloud computing on the other hand started right off with the commercial idea to offer computing resources as a utility to any paying customer. The use of virtualization allows the underlying infrastructure to be abstracted into virtual resource pools [9], [15], [16], [25], [28], [29], [33], [37], [38], though just how abstract depends on the type of cloud offering [2], [9]. Cloud computing is dynamic, flexible and adaptive due to the automatic scalability of the infrastructure [9], [21], [33] and offers unlimited (at least the illusion of unlimited) resources on-demand for the customers [2], [9], [26], [29], [38]. Pay-per-use model has been the dominant payment model of cloud computing [2], [14], [21], [26], [33]. Ease of use is often associated with clouds [14], [21], [26], [29], [33]. All of these aspects conform very closely to the utility computing paradigm. One could say that cloud computing comes closest to fulfilling the utility computing model. Armbrust et al. [2] call cloud computing as the long-held dream of computing as a utility. Sun Microsystem's Cloud Computing primer [28] goes even further and calls cloud computing an evolved form of utility computing that takes it to the next level, as virtually any IT capability can be combined, recombined and delivered in real time [28].

If we think about the terminology, isn't utility computing exactly the same as Infrastructure as a Service (IaaS)? Gartner IT glossary [13] defines IaaS as follows:
"Infrastructure as a service (IaaS) is a standardized, highly automated offering, where compute resources, complemented by storage and networking capabilities are owned and hosted by a service provider and offered to customers on-demand. Customers are able to self-provision this infrastructure, using a Web-based graphical user interface that serves as an IT operations management console for the overall environment. API access to the infrastructure may also be offered as an option."
Unfortunately the Gartner IT glossary doesn't have an entry for utility computing so we can't make a direct comparison from the same source. Foster et al. [9] describe utility computing as:
"a business model in which computing resources, such as computation and storage, are packaged as metered services similar to a physical public utility, such as electricity."
Llorente et al. [19] describe utility computing as:
"a service provisioning model, which will provide adaptive, flexible and simple access to computing resources, enabling a pay-per-use model for computing similar to traditional utilities such as water or electricity. ... The consumer requires a uniform, secure and reliable functionality to access the utility computing service and the provider requires a scalable, flexible and adaptive infrastructure to provide the service."

Clearly both terms are being used to describe the same thing. Wouldn't it be clearer to define cloud computing as the sum of SaaS and IaaS? With all of these "X as a Service", perhaps even a term like Service Computing could be used to describe the whole concept of what clouds (and grids as we will later see) are. Or alternatively we could see SaaS and IaaS both being part of utility computing. Sun Microsystem's Cloud Computing primer [28] talks about cloud computing being an evolved form of utility computing that can deliver virtually any IT capability as a service, including software. This sort of definition would include SaaS as part of utility computing, rather than a separate being. Perhaps these overlapping terms can show the motivation of this paper to try and figure out the terminologies and their relationships better.

In summary, utility computing is a concept that has been discussed about all the way back in the 60s, but the first true implementation of it saw the light of day in the mid 90s with grid computing. The initial idea and goal behind grids was not of commercial nature though. Cloud computing on the other hand started with the motivation of providing computing as a public utility to any paying customer. The different aspects and characteristics of cloud computing fill the utility computing model very accurately. Out of all the terms and paradigms thrown around in discussions about cloud computing, utility computing is the one that most accurately captures what clouds are about. Cloud computing is a concrete manifestation of utility computing, a way to deliver computing as a utility.

4.3 Cloud computing and distributed computing

Distributed computing is another term that is frequently used in discussions about cloud computing. The Gartner IT glossary [13] defines distributed computing as follows:
"A form of computing in which data and applications are distributed among disparate computers or systems, but are connected and integrated by means of network services and interoperability standards such that they function as a single environment."

It's fairly obvious that cloud computing and clouds are a form of distributed computing (Foster et al. [9] call it a "specialized distributed computing paradigm"). At the core of cloud computing are data centers made up of a massive number of commodity hardware connected by a network. On the larger scale, all these massive distributed data centers form the cloud. Going even further would be the federated cloud, which is formed from the distributed clouds. Though using distributed com-

puting to describe cloud computing doesn't really add much to it, since pretty much all modern data centers are based on distributed computing. The grid computing and edge computing paradigms covered later in this chapter are forms of distributed computing as well.

Though the scale and composition of clouds are bit different to grids, edge computing and ordinary data centers. Already back in 2007 Aaron Weiss [34] estimated that Google operated over half a million servers situated at dozen or so physical locations. This makes clouds unique distributed infrastructures as their scale is much larger than normal data centers, yet there are not as many of these massive data centers as the other computing paradigms (grid and edge) often have.

4.4 Cloud computing and grid computing

Grid computing started off in the mid 1990's to create an alternative to supercomputers and large dedicated computing clusters (high performance computing, HPC) available at the time, both of which were expensive and hard to get access to. The concept was to share and make use of resources like computing, storage and networking from multiple geographically distributed institutions and their commodity grade machines. The goal was to offer these shared resources, on-demand, to enable coordinated problem solving and address large-scale computational problems in a dynamic, distributed environment [6], [9], [33]. Institutions would join the grid, offering their own resources for its use, and in turn gaining access to the resources provided by others in the grid. Mechanisms are provided for negotiating resource-sharing arrangements among participating parties (providers and consumers) [8]. Since the resources are provided by multiple different organizations, they are generally heterogeneous and dynamic, as each organization might have different hardware and software platforms and configurations, and the availability and capacity of the resources might vary [9], [19]. "Virtual organizations" (made up of physically distributed institutions and individuals or logically related projects or groups) are formed in the grid, within which distributed resources can be discovered, shared and requested between those who are part of them, in a highly controlled way. A set of standard protocols, middleware, toolkits and services are provided to support these actions [8], [9].

Foster et al. [9] use the following picture (Figure 4.1) to show the relationship between clouds, grids and other related domains such as clusters. Web 2.0 covers the whole range of service-oriented applications, and clouds are at the large-scale side of that domain. On the other side of the spectrum are supercomputers and clusters,

which have been more focused on traditional non-service applications. Grid computing is somewhere in the middle, overlapping all the other domains. Grids are also generally considered to be of lesser scale than clouds or supercomputers [9].

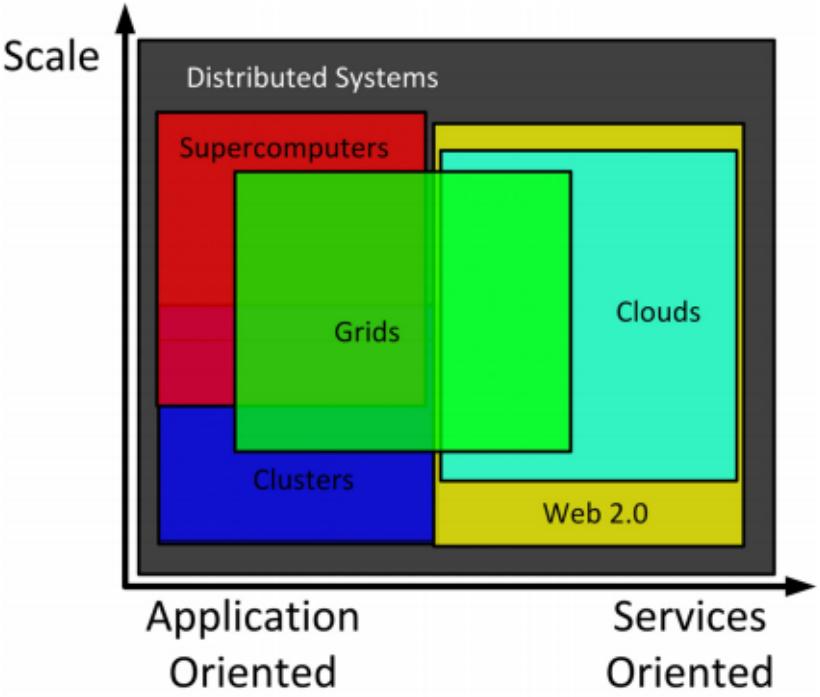

Figure 4.1: Grids and Clouds Overview [9].

Back in 2002 Ian Foster [8] offered a three point checklist to capture the essence of grid systems. According to him, a grid is a system that:

1. *Coordinates resources that are not subject to centralized control...*
2. *... using standard, open, general-purpose protocols and interfaces...*
3. *... to deliver nontrivial qualities of service.*

Vaquero et al. [33] note that more recent definitions of grids concentrate on the combination, management and presentation of the resources from all the different organizations in the grid. Foster et al. [9] note that point 3 of the checklist applies to cloud computing as well, but that points 1 and 2 not so much, at least for today's cloud systems. Next up we take a closer look at clouds and grids with these three points in mind, but we also compare other aspects of them.

Clouds are built, owned and managed by a single massive company. Grid are formed by multiple organizations (or even individuals) joining their resources together to form a bigger entity, the grid. Each organization in the grid are in control of their own infrastructure and resources [6], [8], [9], [36]. Clouds are purely business focused, driven by commercial offers. Grids have traditionally been more academia and government focused, cooperative efforts to share idle resources [9], [14]. Though grid services can be billed as well, typically on a fixed rate per service rather than on the pay-per-use model that clouds use [33]. Perhaps to show the non-profit origin and nature of the grid, no viable commercial grid computing providers emerged until the latter half of 2000's, some ten years after the concept was brought up [9].

The de facto standard implementation for grids is the Globus Toolkit, which provides standard network protocols, middleware and services for building and maintaining a uniform computing environment from a possibly wide range of heterogeneous and diverse resources. The Globus Toolkit deals with issues like security, resource discovery, provisioning and management, job scheduling, monitoring and data management [9], [19]. The Open Grid Services Architecture (OGSA) developed by the Global Grid Forum (GGF) is another initiative that aims to define a common, standard and open architecture for grids [19]. Since grids are formed from multiple distributed organizations, each with their own hardware and software platforms, there have been huge efforts into standardization (of both user interfaces and inner interfaces) in the grid computing community (of which the Globus Toolkit is a good example of) [9], [33]. Clouds on the other hand are proprietary, owned and managed by large corporations. The internal parts of the clouds are rarely built around any standard protocols, interfaces or software and are generally not interoperable with other clouds [9], [28], [33], [36]. Surely clouds being a relatively young computing model has something to do with the lack of standardization efforts [6], but cloud providers might also currently lack business incentives to invest in defining, developing and implementing new interfaces for interaction with other clouds [9]. This is a clear difference between grids and clouds. Grids are heavily standardized and open (and must be so to enable interoperability and a common infrastructure [8], [19]), while clouds are much less so.

As we have already concluded, clouds make heavy use of virtualization to abstract the underlying infrastructure (computing, storage and network resources) into virtual resource pools [9], [15], [16], [25], [28], [29], [33], [37], [38]. Grids do not rely on virtualization as much as clouds do, perhaps due to policy, since each individual organization maintains full control over their resources. The resources in the

grid are not highly abstracted and virtualized as they are in clouds. Though there are efforts to make use of virtualization in grids, trying to achieve many of the benefits that clouds gain from it, especially the dynamic deployment and configuration capabilities [9]. Virtualization could also enhance the scalability and on-demand services of grids [33]. Dillon et al. [6] note that both clouds and grids aim to achieve resource virtualization. It's important to note that this does not necessarily mean using actual virtualization techniques to achieve that goal.

As we have covered already, the business model for clouds is utility computing, a large cloud operator offering computing resources (whether that be hardware or software) and the customer paying for them on a consumption basis. The business model for grids is a bit different, revolving around cooperation of multiple institutions. An organization joins the grid and gains access to all the other resources in the grid, but also gives access to their own resources for other organizations in the grid. Different economy models have been researched for grids that support trading, negotiations, provisioning and allocation of resources based on the levels of services provided, risks, costs and the users' preferences [9]. Services and resources delivered can be customized to meet various qualities of services, such as response time, throughput, availability and security, and multiple resource types can be allocated and combined to meet complex user demands [8]. Some economy models have been proposed and applied in practise, such as resource exchange (e.g. trading storage for compute cycles), auctions, game theory based resource coordination, virtual currencies, resource brokers and intermediaries [9]. The difference between clouds and grids in terms of business model seems to be a single owner making profit with their infrastructure vs. multiple organizations joining together to form an infrastructure (the grid) that can benefit them all (in terms of gaining access to the grid resources and services, and being able to make profit from offering their own resources and services to others).

Grids are typically geared for a batch-schedule compute model, or "job oriented" model [9], [28], [33]. Users submit batch jobs that request certain resources (e.g. 100 processors) for certain time (e.g. 60 minutes). The job would be put in a wait queue until 100 processors would available for 60 minutes, after which the resources would be allocated and dedicated for the job for its duration. This can make the grid a bit stiff, due to expensive scheduling decisions, data staging in and out and potentially long queue times. These qualities make the grid ill-suited for interactive applications with many short-running tasks, although there is research into multi-level scheduling to lessen these problems [9]. Clouds on the other hand use virtualization to share the same physical resources among all the users at the same time, removing

the problems of dedicated resources, the queuing system and the coordination of the services workflow and location. This makes clouds well suited for interactive services and latency sensitive applications [9], [33].

Though these roles are not set in stone, both grids and clouds can support different types of applications and services (loosely coupled or tightly coupled, services or batch-jobs, compute intensive or data intensive, etc) [9]. Virtualization does allow clouds to be more architecture independent and run a variety of applications, while applications for grids need to be ”gridified”, thus imposing harder requirements on the developers [33]. Though this depends heavily on the abstraction level of the cloud offering, e.g. the Google App Engine does heavily limit the type of applications that can be designed for it [2], [9].

Grids are more oriented towards high performance computing. They aim to provide maximum computing capacity for larger tasks. Clouds on the other hand are driven by user needs and business requirements, and are oriented towards dynamic on-demand computing. Clouds are multi-tenant and have many small-to-medium tasks running at the same time [6].

Merging clouds and grids could actually be done either way. A cloud be implemented over existing grid infrastructure and technologies, benefiting from a decade of community efforts in standardization, security, resource management, virtualization [9], monitoring, storage, QoS and federation of different organizations [33]. Or a cloud could join and become a part of a grid. Though this might be bit disproportionate, like an elephant joining to work with ants. The federated cloud, where multiple clouds join together to cooperate and help each other out, would seem more fitting than a cloud joining a grid. The federated cloud could kind of be seen as a grid in itself, just at a much bigger scale. But only kind of, since the concepts have many business and technical differences anyway. In a similar fashion, maybe a very large and business oriented grid aimed at providing utility computing could be seen as a cloud. Instead of being owned by a single company, the grid infrastructure would be provided and governed by multiple smaller organizations. This would of course make things a little more complicated, both in terms of designing the actual grid, the management of it and the decision making, since multiple parties would be in charge of it. In their paper (back in 2006, before cloud computing was even talked about), Llorente et al. [19] talked about ”outsourced grids” that are ”managed by dedicated service providers”. They predicted that as network capacity grows, these outsourced grids will be able to supply resources to businesses and consumers on-demand over the Internet. Sounds a lot like cloud computing? So even if there are clear differences between clouds and grids, maybe these examples show why it can

be easy to confuse the two at the conceptual and higher level.

The programming models for grids and clouds can differ a bit. Programming for grids is complicated by the very thing that is its foundation, the reliance on multiple distributed administrative domains. Each domain has their own hardware and software platforms, resulting in large variations in resource heterogeneity, stability and performance. These resources can also join and leave at any time, creating some challenges to make programs reliable and fault tolerant [9]. Grids commonly use the MPI (Message Passing Interface) parallel programming model and a variety of coordination languages to allow the heterogeneous components to communicate and interact. Distributed and concurrent components are specified and dynamically composed from the variety of resources found in the grid. Most of these languages and systems are integrated into the Globus Toolkit (the de facto, open-source software for building grids) [9].

Clouds tend to use mesh-up's and scripting languages (Java Script, PHP, Python, etc) to control the workflow, since integrating services and applications from various providers would be hard. They take outputs from one service/application, transform them and feed them into another. Clouds generally allow users to access, configure and program services using predefined APIs exposed as web services. Common communication protocols like HTTP and SOAP are used for these services. The underlying systems and services are often proprietary (like the Google App Engine BigTable storage system / GQL query language), making interoperability, integration and movement to other clouds challenging [6], [9], [20], [28], [31], [36].

Both grids and clouds offer automatic scalability, though in a bit different ways. Grid scalability is achieved by increasing the number of working nodes, while clouds resize the virtualized hardware resources. Scalability and self-management is simpler in clouds due to them being owned and managed by a single company, whereas grids face problems for not having a single owner of the whole system. Though in the future federated clouds (multiple clouds working together) could face similar problems [33].

Interoperability and security are the primary concerns of the grid infrastructure, since the resources in the grid can come from different organization with different hardware, software and resource usage policies [9]. Grids deal with these security, privacy, integrity and segregation issues with things like the public-key based GSI (Grid Security Infrastructure) protocols that are used for authentication, communication protection and authorization, and with CAS (Community Authorization Service) designed for advanced resource authorization within and across communi-

ties. Gruber (A Grid Resource Usage SLA Broker) enforces local usage policies and global SLAs, which allows resources at individual sites to be efficiently shared in multi-site, multi-virtual organization environments [9]. As we have already covered in the previous chapter, security and privacy are big concerns in cloud computing as well [6], [9], [17], [20], [29], [31], [37], [38]. For clouds some of these issues are the same (like authentication and authorization), but some are different (like dealing with virtual machine vulnerabilities). Overall security and privacy are concerns for both grids and clouds, and both deal with these issues in their own ways and technologies.

Let's summarize our findings about the similarities and differences of clouds and grids. Note that the things listed here are not absolute, not all of these points are true for every cloud and grid system. Rather the list offers some general guidelines to the differences between the two.

Similarities:

- Both clouds and grids are forms of utility computing. They offer computing resources (CPU, memory, storage, bandwidth, etc) for customers / users. Both can also provide services and applications (SaaS) for customers / users. Both share the same vision of reducing computing costs and increasing reliability and flexibility by moving hardware and software away to be operated by a third party [8], [9], [33].

Differences:

- **Ownership:** Clouds are proprietary, massive dedicated data centers, with one company in charge of all the infrastructure and resources. Grids are owned and operated by a collection of organizations and institutions, each in charge of their own infrastructure and resources [6], [8], [9], [19], [36].

- **Scale and composition:** Clouds are made up of a smaller number of massive data centers. Grids are made up of larger number of geographically distributed and a lot smaller data centers, even individual computers. Overall clouds operate on a much larger scale [9].

- **Infrastructure:** Cloud resources (hardware, software and supporting platforms) are generally more homogeneous. Grid resources are generally more heterogeneous [9], [19]. Clouds deliver more abstract resources and services (through virtualization), while grids deliver more concrete storage and compute resources [9].

- **Business model:** Cloud providers have commercial motivation, they make profit by selling their infrastructure in a pay-per-use model. Grid are formed by cooperation between institutions and organizations (typically academic or government) to provide that infrastructure (the grid) [9], [14]. The motivation is to help each other out by sharing idle resources, though grid resources and services can also be sold, and they are typically billed using a fixed rate per service [33].

- **Applications:** Clouds are geared for loosely coupled, transaction oriented and interactive services / applications (SaaS). Grids are oriented towards tightly coupled batch-schedule work, high performance computing and traditional applications [6], [9].

- **Technology:** Clouds make heavy use of virtualization to create virtual resource pools. Grids use standard protocols, middleware, toolkits and services (e.g. the Globus Toolkit) to control and aggregate the distributed resources in the grid. The differences in the technologies and infrastructures of clouds and grids lead to different solutions for things like security and scalability.

- **Programming models:** Clouds typically use scripting languages and proprietary systems and services. Grids typically use open-source parallel programming models (MPI), coordination languages, software tools (e.g. the Globus Toolkit) and standards [9], [19].

4.5 Cloud computing and edge computing

Edge computing is another interesting computing paradigm that appears to have some connections to cloud computing. It predates cloud computing as it started out and evolved from the Content Delivery Networks (CDNs) founded in the late 1990's [4]. The idea behind the CDNs was to save static web content (such as images and static web pages) to cache servers "at the edge of the Internet" (which leads to the name, edge computing). Desertot et al. [5] write:
"Edge computing is a new computing paradigm designed to allocate on-demand computing and storage resources. Those resources are web cache servers scattered over the ISP (Internet Service Provider) backbones".

So the edge network consists of a large number of distributed cache servers spread across ISP backbones, to be close to the end user. The goal behind these CDNs was to improve response and content delivery times by having the content

cached closer to the end users (the system would use network conditions and other factors to choose the most suitable edge server for the delivery). Other goals were better reliability (due to the large scale and distributed nature of the edge), scalability (due to the dynamic and automatic resource allocation), to reduce network traffic and to put less stress on the service provider's own infrastructure (as the content can be fetched from the closest CDN cache server) [4], [5].

As websites started to become more interactive, the CDNs needed to evolve from the simple act of caching static content [4]. Desertot et al. [5] suggested expanding the edge computing model to allow full outsourcing of applications, including the presentation layer (which was already being done by the CDNs), but also the business logic and data layers. They went on to present their solution to achieve this expanded outsourcing in their paper. Due to the complexities in the management of their proposed architecture, they highlight the importance of an autonomic manager capable of deploying and moving the services dynamically (either creating new instances or replicating existing ones) back and forth between the service provider's own infrastructure and the edge servers, depending on the usage rate. If there is a usage peak on the service, resources should be rented from the edge servers and let them handle some of the workload (Desertot et al. call this the "edge period"). As the number of users decreases, the services should be moved back to the service provider's own infrastructure. The services need to be able to keep their state when moving from one location to another [5].

The system presented by Desertot et al. [5] focuses on dealing with usage peaks rather than fully outsourcing applications and services to be run on the edge servers. The system as such sounds a lot like hybrid clouds – the organization remains in control of their own infrastructure and draw upon public clouds when usage peaks occur. Even the difficulties and challenges of their proposed system are similar to those of hybrid clouds. Both can create difficulty in designing what parts of the applications and services to migrate, what are the dependencies, how to manage the synchronizations and how to distribute the services across the different domains in an effective, secure and problem-free way [5], [27], [28].

The Akamai EdgeComputing distributed application service (I will abbreviate it as AEC to avoid confusion with terms) launched in early 2003 aims to provide the same benefits for interactive applications that CDNs provide for static web content. It aims to provide a globally distributed computing platform where companies can deploy and run their web applications [4]. In their paper Davis et al. [4] from Akamai Technologies call the AEC a form of utility computing or grid computing. The business idea and the act of selling infrastructure resources is indeed utility com-

puting, and the highly distributed infrastructure architecture is a grid in itself. The AEC tries to allow as many parts of applications and services as possible to be outsourced to the edge servers, but does not fully reach this goal as it relies heavily on caching the application data. Some applications simply do not lend themselves to this model, and can require leaving core business logic and transactional databases to the service provider's own data center, while the presentation layer and some parts of these other layers can be moved to the edge [4]. The AEC provides a customer application console that allows users to deploy and monitor their application instances on the edge servers, and it supports multiple programming environments (e.g. J2EE and .NET) and server software (e.g. Apache Tomcat and IBM WebSphere). The AEC does not change the programming model or introduce any proprietary APIs, it only changes the deployment model [4].

The AEC requires splitting the application into an *edge component* and an *origin component*, and each of these are deployed in their corresponding platforms (edge onto the edge, origin onto the service provider's own data center). This division might require redesigning the application. The AEC has a replication subsystem that allows the application's edge components to maintain it's per-user state. Applications with relatively static databases (e.g. product catalogs, site search) are better suited for the edge as they can be easier to cache on the edge servers, while more complex applications (e.g. customer relationship management, online banking) that rely on transactional databases are not well suited for the edge. Deploying these types of applications on the edge requires splitting the application (with the presentation layer going onto the edge) as discussed earlier. Excessive communication between the edge and origin components should be avoided, and if it is necessary, multiple requests should be bundled to reduce the number of communications, and the edge caching should be exploited as much as possible in these requests [4].

As Desertot et al. [5] noted, an edge system requires an autonomic manager due to the complexities of the system. The AEC automatically starts additional application instances when the load on the application increases, and these instances are started on servers close to the users making requests. As the usage peak drops, the additional instances may be automatically stopped [4]. The AEC billing is handled in a pay-per-use model (just like clouds typically are). The exact measurement unit used is requests per month. Applications are given a standard amount of resources for use (e.g. CPU, memory, bandwidth) and additional application resources can be purchased [4].

The AEC also faces some technical challenges, some of which are similar to those found in cloud computing. For one, the AEC is multi-tenant (multiple applications

from many different customers run simultaneously on the same machine), which raises questions about security and privacy. The AEC uses security sandboxes to prevent applications from accessing unauthorized resources and data (and also to prevent over-utilization of the granted resources). Each customer's applications are run in a separate process on each machine [4]. Davis et al. [4] note that their primary concern for security is buggy code, not malicious users. They base this on having a solid business relationship with their customers. This suggests that to be able to deploy software on the AEC the customer needs to go over some contractual arrangement and identification, which according to Armbrust et al. [2] used to be one of the main hindrances for cloud computing.

Other technical challenges edge computing has relate to load balancing, resource management and monitoring, debugging edge applications and application session state. Load-balancing in particular can be challenging in a globally distributed system [4]. The AEC monitors client traffic, network conditions and the applications running on the edge servers, and attempts to balance the work first among the edge server groups, then within them. More instances of edge applications can be started, stopped or moved to other servers within or among the edge server groups. Load-balancing algorithms and agents are used to make this process automatic [4]. The AEC uses a session replication system that allows client session objects to be replicated across the edge servers. The session objects are stored in local caches and compared to the system replicated session object to validate that the session object is "current". The goal is to avoid one centralized database for sessions since it would introduce unnecessary latency [4]. Application sessions are remapped when hardware or software failures occur in the system, network between the user and the server become congested or when load-balancing occurs [4].

The advantages provided by the AEC and other edge computing platforms include on-demand capacity and automatic scalability (extra work is automatically outsourced to the edge servers), better quality of service (better response time due to the service scaling up and the edge resources being closer to the end user, and better fault-tolerance and availability due to the large and distributed nature of the system) and lower costs, effort and risks for the service provider (better utilization rate of the infrastructure since it doesn't need to be over-provisioned to match usage peaks) [4], [5].

Now that we've covered edge computing in more detail, let's take a summarized look at the similarities and differences between cloud computing and edge computing:

Similarities:

- Both cloud computing and edge computing are a form of utility computing, though edge in a more limited form (see differences below). Both offer on-demand resources, automatic scalability and lowered costs, effort and risks in building the infrastructure (can reach higher utilization rate, provisioning is simpler). Both clouds and "edges" are typically owned and operated by a single company (e.g. Akamai, Adero, Mirror Image and Cisco for edge computing). Both are driven by commercial interests and focus on hosting interactive business applications [4], [5], [9].

Differences:

- **Applications:** The utility computing offered by edge computing is more limited since it relies on caching data as much as possible. Applications and services put on the edge servers need to be "cachified", and parts that cannot be cached need to remain on the service provider's own infrastructure [4].
- **Infrastructure:** The edge is made up of smaller edge server groups that are distributed across ISP backbones. Often there are no dedicated "edge centers", just edge servers located inside data centers [5]. Clouds are dedicated, massive data centers. Overall clouds operate on a larger scale.

4.6 Cloud computing and Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is often mentioned in discussions about cloud computing. Another term, Service-Oriented Computing (SOC), is also sometimes used [6], [23]. Mike Papazoglou [23] describes these two as follows:
"Service-Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications/solutions. To build the service model, SOC relies on the Service Oriented Architecture (SOA), which is a way of reorganizing software applications and infrastructure into a set of interacting services."
We will use the term SOA interchangeably with SOC from here on to avoid confusion with these terms. The term SOA seems to be more popular and better known out of these two.

The idea in SOA is that service providers develop services (that can be any functions from simple requests to complicated business processes) and publish them to be used either for clients outside the company over the Internet, or for in-house use

only. These clients can be other solutions, applications, processes or users. Since these services are supposed to be available for a large variety of clients, they must be based on standard (XML-based) languages and protocols to achieve technology neutrality. They need to be loosely coupled (not tied to any client or service side internal structures or conventions), available regardless of the client location and they must provide their service descriptions so that service discovery mechanism are able to find them [23]. Papazoglou [23] further notes:
"Since services may be offered by different enterprises and communicate over the Internet, they provide a distributed computing infrastructure for both intra- and cross-enterprise application integration and collaboration."

So the idea is to have distributed, independent services that can be discovered and invoked as needed. These service are then used as part of developing applications. The SOA model requires some form of inter-service infrastructure to facilitate discovery, interactions and communication between the services [23]. Tsai et al. [32] explore the concept of using clouds as this inter-service infrastructure in their paper. They call the concept Service-Oriented Cloud Computing Architecture (SOCCA). They describe the relationship between SOA and cloud computing as follows:
"SOA and cloud computing are related, specifically, SOA is an architectural pattern that guides business solutions to create, organize and reuse its computing components, while cloud computing is a set of enabling technology that services a bigger, more flexible platform for enterprise to build their SOA solutions. In other words, SOA and cloud computing will coexist, complement, and support each other."

The proprietary, closed in nature of clouds is in stark contrast with the standard, open and technology neutral SOA. The currently available clouds are simply not compatible to serve this sort of purpose. The SOCCA model requires that cloud resources are componentized into different types of independent services (such as storage service, compute service and communication service). These services must have uniform standards and open interfaces so they can be combined with services from other cloud providers [32]. Tsai et al. [32] propose a *Cloud Ontology Mapping Layer* to mask the differences between the different cloud providers and to help migrate applications from one cloud to another. Another layer, the *Cloud Broker Layer*, collects information from the participating cloud providers and publishes it for clients to use. This information includes information about the cloud provider, the resource type and specification and pricing details. It has other features as well, such as ranking the published cloud resources in different categories (e.g. price, availability and security) and it also provides dynamic SLA negotiation. The final and highest layer, the *SOA layer*, allows integration of existing SOA frameworks on

top of the SOCCA model [32].

If we look at both the cloud computing paradigm and SOA, they really don't have much in common with each other. Saying that cloud computing originated or evolved from SOA seems wrong. In fact they almost seem like complete polar opposites: Clouds are proprietary and closed, and do not support interoperability with other clouds. SOA promotes open standards and interfaces and is all about interoperability. That is not to say that cloud computing could become extremely powerful infrastructure for SOA as the SOCCA model shows. But this would require full support for the idea from the cloud providers, and much work in the standardization and interoperability of different clouds. Or perhaps the SOA model or something similar could be implemented inside a single cloud? This sort of solution would bypass all the interoperability problems, and since the clouds are massive in scale, it would still provide a large infrastructure to work with.

Though this is already beside the focus of this chapter. We set out to find the similarities and differences between these paradigms, and I can't see how SOA would be closely related to cloud computing, or how it would have had something to do with the emergence of clouds. The concepts seem incompatible (at least for now), but they could work well together in the future if the cloud providers got on board with a concept like SOCCA.

5 Conclusions

In this paper we went out to find what cloud computing and clouds really are about. To do this we went over a number of definitions given by the industry and researchers. We also looked cloud computing from the perspective of all the actors involved in this computing paradigm, trying to find the motivations, benefits, challenges and even worries that they might have of cloud computing. Finally, we made a side-by-side comparison between cloud computing and other computing paradigms and technologies that are often associated with or mentioned in discussions about cloud computing. We created five research questions that deal with all of these matters, and we will now try to answer them one by one.

1. What is cloud computing? What are clouds? What are the essential characteristics of them?

The definition given by Armbrust et al. [2] of cloud computing (see Figure 2.1) breaks the concept into the bare essentials and makes it easy to understand. Cloud computing at its core is a combination of Software as a Service (SaaS) and utility computing. *Cloud Providers* sell the computing resources of their infrastructure to the *Service Providers* (utility computing), and the *Service Providers* sell their applications and services to the *Service Users* (SaaS). Clouds are the massive data centers (tens or hundreds of thousands of computers each) that form the infrastructure that enables cloud computing. I would try to capture the essential characteristics of cloud computing into the following:

- Availability of infinite amount of abstracted and virtualized resources, on-demand.
- Instant and automatic scalability of the service, resulting in optimal utilization rate.
- Pay only for what you use, reducing or completely eliminating many acquisition, maintenance and operating costs.

2. Why is there confusion about the definition of cloud computing? Is the concept too confusing or broad that this is warranted? Should the definition be redefined or adjusted to make it clearer?

Confusion about cloud computing can come from it being built around many existing technologies and concepts, and from it being a relatively young and hyped computing paradigm. Another possible reason for confusion is the focus on the service provider / service user level of interaction, when the ”new things” that cloud computing enables happen mostly at the cloud provider / service provider level. Concepts like private clouds are also bit sketchy since they miss out on the core features of cloud computing (utility computing, pay-per-use, high utilization and scalability). There's also no official definition of just what exactly is a cloud. Does a data center with 10 000 computers qualify as a cloud? What about 1000? A lot of things are being called clouds these days (possibly for marketing reasons), which further muddies the cloud computing concept. It might be useful to try to create a more specific definition for clouds, at least for academic use. A definition that concentrates on things like the actual scale of the data center, resource utilization efficiency, scalability, cost effectiveness and perhaps even some Quality of Service (QoS) aspects.

3. What are the motives, benefits, concerns and challenges for all the actors in the cloud computing scenario?

Cloud computing allows Infrastructure Providers (the cloud owners) to make money with the massive infrastructure and resources they have built, by offering computing resources as a service to paying customers. This business model is enabled by the economies of scale (large bulk purchases) that few other data centers could leverage. Security is a top priority and challenge for cloud operators due to the multi-tenant nature of clouds. The cloud computing model introduces new security vulnerabilities that need novel ways to deal with. Another important thing for cloud providers are the Service-Level Agreements (SLAs). Since the cloud system is a complex and unpredictable environment, customers require a guaranteed Quality of Service (QoS) to gain confidence to move their applications into the cloud. The SLAs deal with things like availability and performance of the service, and these goals are monitored and enforced by autonomic systems.

Service Providers (the companies who deploy software into the cloud) can gain numerous advantages from using clouds. These include reduced (or completely removed) costs on hardware acquisition, maintenance, software licenses, upgrades,

employees, facilities and power. It allows paying only for the resources that are used, and in general reduces costs, effort and risks since the company's own infrastructure doesn't need to be over-provisioned and thus reaches higher utilization rate. A service deployed in a cloud can instantly scale up (and down) to meet peaks in demand, ensuring better quality of the service. Using clouds can also speed up application development, deployment, and shorten the time to market. Small companies without their own infrastructure can deploy a service in clouds in a quick, cheap and risk-free way.

The main concern for service providers are the security and privacy risks cloud computing raises since many applications and services from multiple customers run on the same physical hardware. Other problems include migrating legacy applications to the cloud and the inability to move applications from one cloud provider to another (which can put the service provider "at the mercy" of the cloud provider). Cloud computing also presents some challenges for the software engineering process (making it more complex) and can seem threatening to the open-source community.

The advantages of cloud computing for Service Users (the end users) are more mobile and accessible applications and reduced effort in software and hardware management. The fears of security and privacy issues are the same for service users as they are for service providers. Loss of control and being at the mercy of the cloud / service provider can raise concerns for advanced users.

4. What are the differences and similarities between the variety of technologies, computing models and paradigms that are often associated with cloud computing? How exactly are they related to clouds?

Software as a Service (SaaS) is an integral part of cloud computing, as any applications or services deployed in a cloud qualify as SaaS. It is the other main half of what makes up cloud computing. Utility computing is the other half. Utility computing is the essence of cloud computing – delivering computing resources as a utility. Cloud computing is the first computing model to fully embrace and deliver quick and easy utility computing to any paying customer. Cloud computing is based on distributed computing, but so are most of the other related computing paradigms and ordinary data centers. Using distributed computing specifically to describe cloud computing can thus be a wasted effort. Grid computing is another computing model that delivers utility computing. It predates cloud computing and isn't as business oriented as clouds. Grids and clouds have many differ-

ences, such as ownership: Clouds are owned and operated by a single company, grids are formed from infrastructures owned and operated by multiple different organizations and institutions. There are other differences in the scale and composition of the cloud and grid infrastructures, types of applications they are geared for, in the underlying technologies, standardization efforts and programming models.

Edge computing is yet another form of utility computing. It evolved from the simple Content Delivery Networks (CDNs) to a platform where companies can deploy and run web applications. Though the utility computing the edge offers is bit more limited, as it relies heavily on caching data. More complex and dynamic applications and services might not lend themselves to this model, and only parts of them can be outsourced to the edge. The edge infrastructure is made up of edge server groups distributed across ISP backbones, and there often aren't dedicated "edge centers", which contrasts the small number of massive dedicated data centers that make up the clouds. SOA is a computing paradigm that involves creating and publishing services, and applications and solutions are then built using these distributed services. The paradigm promotes open standards and interfaces to make interoperability between different service providers possible. SOA and Cloud computing are not really related to each other, in fact the closed in and proprietary nature of clouds is in stark contrast with the SOA principles. SOA deals with software development using services, cloud computing is about providing computing resources as a utility.

5. Can cloud computing be considered an independent new paradigm, or is it just a combination of existing approaches, concepts and technologies? Which of the features make it stand out from the other computing models?

Based on the previous answers and overall on this paper, I would say cloud computing does stand out as a completely new computing paradigm. It deserves to be its own paradigm, at least as much as grid computing and edge computing do. Cloud computing combines the SaaS and utility computing concepts to offer a fully service oriented platform. The main features that are unique to cloud computing are the *quick* and *easy* access for *any paying customer* to unlimited abstracted resources (which could be pretty much any IT capability), the instant and automatic scalability (a grid, an edge or an ordinary data center simply cannot match the scale and power of the cloud) and the pay-per-use model of the resources (the grid and edge can offer pay-per-use model as well, so perhaps this aspect is not totally unique). Also the underlying infrastructure and technologies used in these paradigms are very different. Though in the future these paradigms might "learn" and adopt features from each

other and evolve to become closer to each other. For example purely business oriented grids have started to emerge in recent years, and clouds could learn a thing or two about standardization efforts from the grid community (which is likely to happen when cloud interoperability becomes more important in the future, e.g. in the form of federated clouds or with SOA gaining popularity among cloud providers).

References

[1] Amazon Web Services, Amazon EC2 Pricing, <URL: http://aws.amazon.com/ec2/pricing/>. Referenced in 1.4.2013.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, *Above the Clouds: A Berkeley View of Cloud Computing*. Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2009-28. A published version available at <URL: http://dl.acm.org/ft_gateway.cfm?id=1721672&ftid=757661&dwn=1&CFID=217788895&CFTOKEN=71288283>.

[3] BBC Technology News, *OnLive's video gaming tech was sold for less than $5m*, 10 October 2012. Available at <URL: http://www.bbc.co.uk/news/technology-19896362>. Referenced in 11.4.2013.

[4] Andy Davis, Jay Parikh, William E. Weihl, *EdgeComputing: Extending Enterprise Applications to the Edge of the Internet*. Proceedings of the 2004 13th international World Wide Web conference on Alternate track papers posters (WWW Alt. 2004), p. 180-187. Available in PDF at <URL: http://dl.acm.org/ft_gateway.cfm?id=1013397&ftid=277916&dwn=1&CFID=331436560&CFTOKEN=41154957>.

[5] Mikael Desertot, Clement Escoffier, Didier Donsez, *Autonomic Management of J2EE Edge Servers*. Proceedings of the 2005 3rd international workshop on Middleware for grid computing (MGC 2005), p. 1-6. Available in PDF at <URL: http://dl.acm.org/ft_gateway.cfm?id=1101503&ftid=337851&dwn=1&CFID=331436560&CFTOKEN=41154957>.

[6] Tharam Dillon, Chen Wu, Elizabeth Chang, *Cloud Computing: Issues and Challenges*. Published in 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), p. 27-33. Available in PDF at <URL:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5474674>.

[7] Vincent C. Emeakaroha, Marco A.S. Netto, Rodrigo N. Calheiros, Ivona Brandic, Rajkumar Buyya, César A.F. De Rose, *Towards autonomic detection of SLA violations in Cloud infrastructures*. Future Generation Computer Systems, Volume 28, Issue 7, July 2012, p. 1017-1029, Special section: Quality of Service in Grid and Cloud Computing. Available in PDF at <URL: http://www.sciencedirect.com/science?_ob=MiamiImageURL&_cid=271521&_user=1234512&_pii=S0167739X11002184&_check=y&_origin=article&_zone=toolbar&_coverDate=2012-Jul-31&view=c&originContentFamily=serial&wchp=dGLzVBA-zSkzV&md5=8ffb1777976add27505000065b5aae9d&pid=1-s2.0-S0167739X11002184-main.pdf>.

[8] Ian Foster, *What is the Grid? A Three Point Checklist*. GRIDtoday, Volume 1, Number 6, July 2002. Available in PDF at <URL: http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf>.

[9] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu, *Cloud Computing and Grid Computing 360-Degree Compared*. Published in 2008 Grid Computing Environments Workshop (GCE 2008), p. 1-10. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4738445>.

[10] Marc Fouquet, Heiko Niedermayer, Georg Carle, *Cloud Computing for the Masses*. Proceedings of the 2009 1st ACM workshop on User-provided networking (U-NET 2009): Challenges and opportunities, p. 31-36. Available in PDF at <URL: http://dl.acm.org/ft_gateway.cfm?id=1659038&ftid=711155&dwn=1&CFID=217788895&CFTOKEN=71288283>.

[11] Gamers Bliss, Kasey Milinkovich, *Has SimCity and Diablo III Ruined Always-Online DRM?*, 10 March 2013. Available at <URL: http://www.gamersbliss.com/2013/03/10/have-simcity-and-diablo-iii-ruined-always-online-drm-forever/>. Referenced in 18.4.2013.

[12] Games Industry, Mike Williams, *PlayStation 4 to use Gaikai for backwards compatibility*, 16 February 2013. Available at <URL: http://www.gamesindustry.biz/articles/2013-02-16-playstation-4-to-use-gaikai-for-backwards-compatibility>. Referenced in 12.4.2013.

[13] Gartner IT Glossary, <URL: http://www.gartner.com/it-glossary/>. Referenced in 8.5.2013.

[14] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen and Zhenghu Gong, *The Characteristics of Cloud Computing*. Published in 2010 39th International Conference on Parallel Processing Workshops (ICPPW 2010), p. 275-279. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5599083>.

[15] Bernd Grobauer, Tobias Walloschek, Elmar Stöcker, *Understanding Cloud Computing Vulnerabilities*. Published in Security Privacy, IEEE, Volume 9, Issue 2, p. 50-57, 2011. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5487489>.

[16] Radha Guha, David Al-Dabass, *Impact of Web 2.0 and Cloud Computing Platform on Software Engineering*. Published in 2010 International Symposium on Electronic System Design (ISED 2010), p. 213-218. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5715178>.

[17] Brian Hayes, *Cloud Computing*, Communications of the ACM, Volume 51, Issue 7, July 2008. Web science, Column: News, p. 9-11, 2008. Available in PDF at <URL: http://dl.acm.org/ft_gateway.cfm?id=1364786&ftid=544378&dwn=1&CFID=217788895&CFTOKEN=71288283>.

[18] Eric Keller, Jakub Szefer, Jennifer Rexford, Ruby B. Lee, *NoHype: Virtualized Cloud Infrastructure without the Virtualization*. Proceedings of the 2010 37th annual international symposium on Computer architecture (ISCA 2010), p. 350-361. Available in PDF at <URL: http://dl.acm.org/ft_gateway.cfm?id=1816010&ftid=812584&dwn=1&CFID=217788895&CFTOKEN=71288283>.

[19] Ignacio Llorente, Rubén Montero, Eduardo Huedo, Katia Leal, *A Grid Infrastructure for Utility Computing*. Published in 2006 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2006), p. 163-168. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4092200>.

[20] Rich Maggiani, *Cloud Computing Is Changing How We Communicate*. Published in 2009 IEEE International Professional Communication Conference (IPCC 2009), p. 1-4. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5208703>.

[21] Dave Malcolm, *The five defining characteristics of cloud computing*, 2009. Available at <URL: http://www.zdnet.com/news/the-five-defining-characteristics-of-cloud-computing/287001>. Referenced in 4.11.2012.

[22] Michael Maurer, Ivona Brandic, Rizos Sakellariou, *Adaptive resource configuration for Cloud infrastructure management. Future Generation Computer Systems*, Volume 29, Issue 2, February 2013, p. 472â487, Special section: Recent advances in e-Science. Available in PDF at <URL: http://www.sciencedirect.com/science?_ob=MiamiImageURL&_cid=271521&_user=1234512&_pii=S0167739X12001525&_check=y&_origin=article&_zone=toolbar&_coverDate=2013-Feb-28&view=c&originContentFamily=serial&wchp=dGLzVBA-zSkzV&md5=8d8f4c95aaa59f73358499562228c38c&pid=1-s2.0-S0167739X12001525-main.pdf>.

[23] Mike P. Papazoglou, *Service -Oriented Computing: Concepts, Characteristics and Directions*. Proceedings of the 2003 Fourth International Conference on Web Information Systems Engineering (WISE 2003), p. 3-12. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1254461>.

[24] Polygon News, Michael McWhertor, *PS4 to use Gaikai for instant demos, Ustream multicasting, social gameplay*, 20 February 2013. Available at <URL: http://www.polygon.com/2013/2/20/4006146/playstation-4-streaming-ps3-games-playstation-cloud-gaikai>. Referenced in 11.4.2013.

[25] Sameer Rajan, Apurva Jairath, *Cloud Computing: The Fifth generation of Computing*. Published in 2011 International Conference on Communication Systems and Network Technologies (CSNT 2011), p. 665-667. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5966533>.

[26] Rajiv Ranjan, Rajkumar Buyya, Manish Parashar, *Special section on autonomic cloud computing: technologies, services, and applications*. Concurrency and Computation: Practice and Experience, Volume 24, Issue 9, p. 935â937, 25 June 2012. Available in PDF at <URL: http://onlinelibrary.wiley.com/doi/10.1002/cpe.1865/pdf>.

[27] Prashant Sharma, *Characteristics Of Cloud Computing*, 2009. Available at <URL: http://www.techpluto.com/cloud-computing-characteristics/>. Referenced in 4.11.2012.

[28] Sun Microsystems, *Taking Your Business to a Higher Level*, 2009.

[29] Hassan Takabi, James B.D. Joshi, Gail-Joon Ahn, *Security and Privacy Challenges in Cloud Computing Environments*. Published in Security Privacy, IEEE, Volume 8, Issue 6, p. 24-31, December 2010. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5655240>.

[30] The Guardian, Games Blog, Keith Stuart, *Why OnLive's brave venture failed*, 21 August 2012. Available at <URL: http://www.guardian.co.uk/technology/gamesblog/2012/aug/21/what-happened-to-onlive>. Referenced in 11.4.2013.

[31] Alok Tripathi, Abhinav Mishra, *Cloud Computing Security Considerations*. Published in 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), p. 1-5, 2011. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6061557>.

[32] Wei-Tek Tsai, Xin Sun, Janaka Balasooriya, *Service-Oriented Cloud Computing Architecture*. Published in 2010 Seventh International Conference on Information Technology: New Generations (ITNG 2010), p. 684-689. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5501650>.

[33] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, Maik Lindner, *A Break in the Clouds: Towards a Cloud Definition*, ACM SIGCOMM Computer Communication Review, Volume 39, Issue 1, January 2009. Session: Editorial zone, p. 50-55, 2008. Available in PDF at <URL: http://dl.acm.org/ft_gateway.cfm?id=1496100&ftid=606313&dwn=1&CFID=217788895&CFTOKEN=71288283>.

[34] Aaron Weiss, *Computing in the clouds*, netWorker, Volume 11, Issue 4, December 2007. Cloud computing: PC functions move onto the web, p. 16-25, 2007. Available in PDF at <URL: http://di.ufpe.br/~redis/intranet/bibliography/middleware/weiss-computing08.pdf>.

[35] Windows Azure, Purchase options, <URL: http://www.windowsazure.com/en-us/pricing/purchase-options/>. Referenced in 1.4.2013.

[36] Dong Xu, *Cloud Computing: an Emerging Technology*. Published in 2010 International Conference on Computer Design and Applications (ICCDA 2010), Volume 1, p. 100-104. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5541105>.

[37] Jianfeng Yang, Zhibin Chen, *Cloud Computing Research and Security Issues*. Published in 2010 International Conference on Computational Intelligence and Software Engineering (CiSE 2010), p. 1-3. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5677076>.

[38] Shuai Zhang, Shufen Zhang, Xuebin Chen, Xiuzhen Huo, *Cloud Computing Research and Development Trend*. Published in 2010 Second International Conference on Future Networks (ICFN 2010), p. 93-97. Available in PDF at <URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5431874>.