TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Simo-Pekka Leppänen

Lightweight Method for Evaluating Cloud Compatibility

Master of Science thesis

Examiner: Prof. Tommi Mikkonen
Supervisor: Henri Laamanen, M.Sc.
Examiner and topic approved by the
Council of the Faculty of Computing
and Electrical Engineering on
12th of August 2015

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
Simo-Pekka Leppänen: Lightweight Method for Evaluating Cloud Compatibility
Master of Science Thesis, 52 pages
June 2016
Major: Software Engineering
Examiner: Prof. Tommi Mikkonen
Supervisor: Henri Laamanen, M.Sc.

Keywords: Cloud computing, XaaS, Microservices

Cloud services have gained popularity in the past few years, and many companies are offering their software as a service. Cloud environments offer scalability, and it is indeed easy to start using a cloud service instead of acquiring the required hardware. However, some architectural patterns are better in a cloud environment than others. Business critical software that has existed for a long time, such as the operations and business support systems (OSS/BSS) of telecommunication operators, may require extensive changes in order to stay competitive and gain the benefits of cloud environments. The number of mobile device and Internet users continues to grow, and the scalability provided by cloud environments could help OSS/BSS systems handle the growing load.

This thesis focuses on the opportunities that cloud provides, and problems faced by companies looking for ways to move their mature products to a cloud environment. Moving software from customer premises to a cloud introduces security and latency problems, but would offer benefits with scalability, if the legacy software can be transformed to a cloud compatible architecture, such as microservices architecture. Such a transition also affects the way the software is developed and how it is deployed.

The result of this thesis is a method for evaluating the cloud compatibility of a software product. The method was also used to evaluate the feasibility of deploying Comptel InstantLink to a cloud environment. The architecture of Comptel InstantLink requires changes so that it could be automatically scaled. However, cloud environments would provide value to the users of Comptel InstantLink. A private cloud environment deployed to the telecommunication operator's own infrastructure would be a suitable environment for Comptel InstantLink. The method created in this thesis proved to be a useful starting point for evaluating cloud compatibility, and helps detecting the main areas of concern in cloud migration.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
Simo-Pekka Leppänen: Kevyt menetelmä ohjelmistojen pilviyhteensopivuuden arvioimiseen
Diplomityö, 52 sivua
Kesäkuu 2016
Pääaine: Ohjelmistotuotanto
Tarkastaja: Prof. Tommi Mikkonen
Ohjaaja: Henri Laamanen, M.Sc.

Avainsanat: Pilvipalvelut, XaaS, Mikropalvelut

Pilvipalveluiden suosio on kasvanut viime vuosina, ja monet yritykset tarjoavat ohjelmistojaan pilvipalveluina. Pilviympäristöt tarjoavat mahdollisuuden skaalautuvuuteen, ja niitä on helppo alkaa käyttää sen sijaan, että hankkisi tarvittavan laitteiston omiin tiloihin. Ohjelmistoja, jotka ovat liiketoiminnan kannalta kriittisiä, ja jotka ovat olleet olemassa jo kauan, voi olla vaikea siirtää pilviympäristöön. Ne voivat vaatia merkittäviä muutoksia, jotta pilviympäristön ominaisuuksista saadaan hyötyä. Esimerkki tällaisista ohjelmistoista ovat teleoperaattorien liiketoiminnan tukiohjelmistot. Mobiililaitteiden ja Internetin käyttäjämäärät jatkavat kasvua, ja tukiohjelmistojen täytyy kyetä käsittelemään yhä suuremmat määrät tilauksia. Pilviympäristöt voivat olla hyvä vaihtoehto liiketoiminnan tukiohjelmistoille skaalautuvuuden parantamiseksi.

Tämä diplomityö keskittyy pilviteknologioiden tarjoamiin hyötyihin sekä ongelmiin, joita yritykset kohtaavat halutessaan siirtää ohjelmistonsa pilveen. Ongelmiksi muodostuvat varsinkin tietoturva ja suorituskyky. Pilvessä skaalautuvuus olisi hyvä, jos ohjelmiston vanha arkkitehtuuri saataisiin muutettua pilviyhteensopivammaksi, esimerkiksi käyttäen mikropalveluarkkitehtuuria. Muutos vaikuttaa myös siihen, kuinka ohjelmistoa kehitetään ja kuinka se toimitetaan.

Työn tuloksena esitellään menetelmä pilviyhteensopivuuden mittaamiseksi. Menetelmää käytettiin myös arvioimaan Comptel InstantLinkin pilviyhteensopivuutta. Pilvestä olisi mahdollista saada hyötyä, mutta Comptel InstantLinkin arkitehtuuria tulee muuttaa hyötyjen saavuttamiseksi. Comptel InstantLink sopisi parhaiten asennettavaksi yksityiseen pilviympäristöön teleoperaattorien sisäverkossa. Työssä esitelty menetelmä osottautui hyödylliseksi lähtökohdaksi pilviyhteensopivuuden arvioimisessa. Sen avulla on helppo löytää mahdolliset ongelmakohdat pilviympäristöön siirryttäessä.

PREFACE

This thesis was done while working as Junior Software Engineer at Comptel in Helsinki. I am grateful for being given the opportunity to work on my thesis during working hours. I would like to thank my colleagues for being so supportive and for the great working environment. I was even forced to concentrate on the thesis, if I tried to contribute on some other work during a thesis day.

Special thanks go to the examiner of my thesis, professor Tommi Mikkonen, and my supervisor Henri Laamanen. Your feedback was always very useful and I was happy to receive it always at such a short notice. Thanks to my sister Anu for proof-reading.

Finally, I would like to thank Mia and my friends for all the support during times of wavering confidence.

Helsinki, May 19, 2016

Simo-Pekka Leppänen

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface, an interface which describes what kind of operations an application provides, what inputs it requires, and what are the outputs.
IaaS	Infrastructure as a Service, a cloud service model where customers rent virtual environments in a highly scalable computing infrastructure and pay for usage.
I/O	Input/Output, hard drive I/O operations are reading and writing.
Jitter	Variability in latency.
Latency	Time it takes to receive a response after sending a request to a web server.
NE	Network Element, a device in a communication service providers network.
NEI	Network Element Interface, an interface that Comptel InstantLink uses for sending tasks to network elements.
NFV	Network functions virtualization, the virtualization of specialized network devices so their functions can be run on commodity hardware.
OSS/BSS	Operations and Business Support Systems, systems that automate the work of telecommunication service providers.
PaaS	Platform as a Service, a cloud service model where a development and deployment environment is offered as a service.
RPE	Request Processing Engine, a component in Comptel InstantLink that handles incoming requests.
SaaS	Software as a Service, a cloud service model where software is offered as a web application.
SDN	Software defined networking, the separation of the control and data planes of networking devices.

SLA	Service level agreement, an agreement containing details of the service provided by a cloud provider.
SME	Service Module Engine, a component in Comptel InstantLink that is used to add modules.
XaaS	Anything as a Service, a notation where X can be replaced with something to highlight that the service is running in a cloud environment. For example, Big Data as a Service.

1. INTRODUCTION

The Internet has changed our daily life in many ways, and the change is not over yet. Two decades ago we started looking for information from web pages, and today services are available for us throughout the day. Simple applications are just for checking bus timetables and reading email while more and more complex software is available to us as services through web sites. Cloud environments and virtualization can be good ways for companies to save on infrastructure costs and maintain a good quality of service even during high peaks of usage.

Cloud environments are usually advertised as the right way to go and something that everyone will be using in the future. To gain the benefits that a cloud environment has to offer, a company has to carefully measure costs and security risks that are very different from an on-premise deployment of a traditional client-server application. The benefits cannot be obtained if the deployed software does not meet certain criteria concerning scalability and resource needs. Cloud environments are often claimed to reduce the costs caused by infrastructure. However, it is not that straight forward, because large amounts of data and network traffic can be expensive.

There are existing frameworks for assessing cloud readiness of software that a company uses and others for assessing the products created by the company. However, these frameworks are often complex. An easy way to determine whether it makes sense to deploy own software products to a cloud environment is required. That would be useful especially since the problems of cloud environments are not that well known but everyone wants to be part of the latest technology trend.

This thesis presents a lightweight flow diagram based method for assessing the cloud readiness of a software product as a result of an integrative literary review. Chapter 2

introduces cloud environments and different kinds of service levels. Chapter 3 presents key factors affecting cloud deployments and flow diagram questionnaires for assessing cloud readiness. The framework was used in a case study to evaluate the cloud readiness of a provisioning software product and the case study is presented in Chapter 4. The results of the study and the created method are evaluated in Chapter 5.

2. CLOUD COMPUTING

In this chapter we will take a look at the definition of cloud computing. In Section 2.1. we discuss the history of different computing environments and what makes clouds different. Ownership and access to a cloud is defined as deployment models, which are presented in Section 2.2. Service models presented in Section 2.3. define the level of abstraction in a cloud environment, and the typical ways of pricing used for these models are presented in Section 2.4. The benefits and problems of cloud environments are discussed in Sections 2.5. and 2.6.

2.1. Evolution of Computing Environments

Earliest visions of computing as a utility in the same way as water, electricity, gas, and telephony date back to the 1960s. The networks were still young, but it was predicted that some day computer utilities would be used as a service. Mainframe computers can be considered to resemble utility computing, because one large computer offered computing utilities as a service to the end users sitting in front of a terminal. Personal computers were a step away from the utility way of thinking, but the computing power was brought to the homes of many users. The Internet gave users new ways to fetch information and use services, and the client-server model moved some of the functionality away from the personal computers and to the servers. [1]

In the 1990s grid computing emerged to help researchers obtain the computing power they needed. The computers in the grid can be geographically far from each others and are often, for example, the workstations or personal computers of other people. The name grid came from power grids and the way they offered resources to homes. On some level, grid computing can be seen as something that the visionaries of the 1960s were dreaming

of but it did not bring computing as a utility for the masses, as it is used for specific types of computing. [1]

Cloud computing has emerged in the late 2000s. There has been confusion regarding what cloud computing actually is, and even claims that it is only a marketing name for old technologies [2]. From the service user's point of view cloud computing is just a client-server architecture. The user uses some service that can be accessed through a web page the same way as was done already in the 1990s although the service can be much more complex nowadays. [3]

However, cloud computing does have some distinct characteristics that are not visible to the end users. A cloud is a shared collection of computing resources including CPUs and storage devices. Computers in the cloud are virtualized and the actual hardware can be completely hidden even from the cloud users that utilize the hardware in order to provide web services. The computation resources can be rapidly and even automatically scaled up and down depending on the need, and cloud users only pay for what they are using instead of a fixed monthly price. The features differentiating cloud computing from a traditional client-server architecture are virtualization, scalability, service orientation, and the payment options. The services can be from many abstractions levels including hardware, platforms, and software. [3]

2.2. Deployment Models

Cloud deployment models divide cloud environments into distinct classes. The models are distinguished mostly by ownership and access. There are four different cloud deployment models: private, community, public, and hybrid cloud [3].

A private cloud is a cloud infrastructure used by a single company. The infrastructure can be maintained and the hardware owned by the company or a third party. A private cloud can be used to make sure that sensitive data is stored according to regulations or to gain better performance for applications that perform a lot of disk input/output (I/O) operations. The trade-off is that the infrastructure costs may have to be paid by the company utilizing the cloud. [3]

A community cloud is shared by companies that have certain shared goals or requirements. Similarly to private clouds, it is also owned and maintained either by some of the using organizations or a third party. [3]

Public clouds are maintained by a cloud provider who owns the hardware. Public clouds can be used by any company or person who buys the service. The benefit of a public cloud is that users do not need to worry about maintaining the hardware and many features are offered as a service by the provider. The provider can, for example, handle all the necessary work to provide load balancing and automatic scaling. [3]

Hybrid clouds are a combination of the previous deployment models. Companies working in, for example, the healthcare field could utilize a community cloud for storing and handling highly regulated data but also use a public cloud for compute intensive analysis. [3]

Different deployment models have varying benefits and different kinds of costs. If the application does not handle highly sensitive data and is not affected by latency and varying performance, a public cloud is the best deployment model. Public clouds offer a lot of flexibility in the form of per usage pricing and different kinds of features that can be bought from the provider. If a steady throughput and low latency is a requirement, a private cloud is a better option because issues caused by the network and other users can be avoided. A hybrid solution is also a good option since some of the business critical functionality can be handled or data stored in a private or community cloud while end users access the application through a service running in a public cloud. The service running in the public cloud can then benefit from scalability, making sure the end users get a smooth and responsive service.

2.3. Service Models

A cloud provider can offer different kinds of services to the cloud users. The most common service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). There are also other models that use the notation Anything as a Service (XaaS), for example, Big Data as a Service (BDaaS). The National

Institute of Standards and Technology only mention IaaS, PaaS, and SaaS as the service models in their definition of cloud computing [3]. In theory, the other XaaS models could be categorized under the three, but in a business context, the notation is useful for highlighting the purpose of the service and that it is deployed to a cloud.

The users and providers of cloud computing can be divided to cloud providers, cloud users, and SaaS users. The cloud provider is the organization that owns the hardware and offers different IaaS and PaaS solutions to cloud users. Cloud users, who can also be SaaS providers, utilize the hardware services to deploy their own applications for internal use or to offer their software as a service. SaaS users are the end users using a web-based application. Cloud users can also be SaaS users if the service that they provide uses another service. [2]

IaaS users gain access to virtualized computing resources on which software can be installed and run. The users can manage the operating system, applications and possibly some networking components such as firewalls. The provider is responsible for the virtualization, hardware and data storage. One example is the Amazon EC2 [4] where the users can manage their virtual machines. The user can select a templated Amazon Machine Image or even their own Virtual Machine Image, which can already contain everything needed or function as the basis on which the environment is built. IaaS can be used to create a customized environment for applications with the added benefit of highly scalable resources. [3]

PaaS users are able to deploy their own applications to the service provider's cloud. They can only manage the applications and the data related to them but not the underlying infrastructure, such as the operating system. For example, developers can deploy their applications to Google App Engine [5] or Microsoft Azure [6]. These kinds of platforms often support only certain programming languages and offer additional services such as storage, databases, and usage monitoring with easy to use APIs or graphical user interfaces. They have different kinds of payment options and also allow the user to make profit with their deployed applications. Many platforms offer a free version with limited resources so that developers can easily do proof-of-concept demonstrations and see if the environment suits their needs. [3]

SaaS users use a software provided by the cloud provider or, for example, a PaaS user. They only have access to possible user account related settings, such as email filters and folders in the case of an email client. Microsoft offers Office 365 where the user can use office applications and email, and save documents and other files to a storage provided through the service. Microsoft Office has long been available as standalone programs but the cloud environment provides the users access to their documents and files from any device connected to the Internet. [3]

2.4. Sales Models

One of the characteristics of cloud computing is usage-based pricing. Cloud providers charge for CPU power, storage space, and networking but also offer more traditional rental services of hosts and computing power. Many providers offer discounts based on subscription types and lengths and even usage. In Subsection 2.4.1. we will focus on different pricing models for scalable cloud environments and in Subsection 2.4.2. we will take a look at different software licenses and how they fit cloud environments.

2.4.1. IaaS and PaaS Offerings

Cloud providers have highly varying pricing models for IaaS and PaaS. What the popular IaaS providers Amazon, Google, and Microsoft have in common is that they offer different types of virtual machine instances with a certain amount of CPU cores, memory, and storage space. Often the price is stated per hour but, for example, Amazon Elastic Compute Cloud (Amazon EC2) instances can also be reserved for a fixed time prediod. Other costs come from network traffic, security, and separate data storage services. PaaS offerings have similarities to IaaS offerings. Prices are often dictated by the type of instances that will run the deployed application, but extra services, for example different kinds of storage options, can also be bought. [4][6][7]

Amazon EC2 instances can be On-Demand Instances which have an hourly rate that is charged when the instance is running. These instances are optimal when the usage of the deployed application varies highly and there can be periods during which the application

is not used at all. Reserved Instances are reserved for a time period and can be paid monthly, partly upfront, or completely upfront. The size of the upfront payment and type of the instance defines a discount. This discount can be further applied by using Amazon's Auto Scale service which provisions On-Demand Instances based on need and applies the same discount for all On-Demand and Reserved Instances of the same type. By combining Reserved and On-Demand Instances, cloud users can make sure they have enough resources for continuous load but are also prepared for peak times. Users can also bid on unused instances called Spot Instances. If the bid exceeds the current Spot price, the Spot Instance will be utilized as long as the Spot price is lower than the user's configured maximum price, or until the instance is terminated by the user. Amazon also offers discounts for network traffic so that the more data is sent out from the cloud the less it costs per gigabyte. AWS Elastic Beanstalk is an additional service, which functions as a platform. Users can upload their code, and Elastic Beanstalk handles deployment, load balancing, and scaling. It is free to use on Amazon EC2 instances, and the user can still access the instance like any other EC2 instance. [4][8]

Google's Compute Engine instances can be selected from predefined types or be customized and have a per minute pricing based on the performance of the instance. Google does not offer monthly subscription. The per minute price is discounted incrementally based on monthly usage up to an effective discount of 30%, when the instance is used throughout the month. Google App Engine is a PaaS, which uses the same kind of pricing for resources as the Compute Engine. The user does not, however, gain access to the instance and can only manage the deployed application through management interfaces. The platform offers many different APIs that can be used, for example, for storing images, creating task queues and managing users. [5][7]

Microsoft Azure offers many different kinds of environments for specialized needs. The most basic IaaS and PaaS offerings are Virtual Machines, Cloud Services and App Service. Virtual Machines is a typical IaaS where the user is responsible for all the configuration. The virtual machine images can be provided by Microsoft or the user can deploy their own images. Cloud Services allows the user to, for example, configure the amount of running services but the provider is responsible for updating the operating sys-

tem. App Service is the easiest option. It allows users to deploy their applications while the provider handles the administrative work. Virtual Machines and Cloud Services have similar pricing as Amazon's and Google's virtual machine instances. The price depends on the performance of the instance and selected support plan. App Service has different service plan levels which determine how many applications and instances can be running and how much storage space can be used. In addition, each service level has instance types that have an hourly and monthly price based on their performance. [6]

Some cloud providers also offer private cloud services. The provider is responsible for maintaining the cloud infrastructure and, for example, Rackspace offers options to run the cloud in their data center or in the customer's own data center. Rackspace's private cloud software is open source and free which also gives the option to easily run a private cloud without giving anyone else access to the environment. This way companies requiring tight security can gain the benefits of efficient scaling and load balancing but will not be able to save on hardware and maintenance costs. [9]

2.4.2. Software Licenses

Pricing models for SaaS vary from traditional licenses to usage based pricing. Traditional perpetual licenses are large upfront payments, which grant the customer full usage rights and certain level of support and maintenance. Committing to a license that costs the same as 3-5 years of subscription payments is a difficult decision to make since it prevents the customer from changing their service. [10]

A more flexible approach is a tiered pricing model. Higher tiers can offer more capacity in the form of more user accounts, storage space or, for example, version control repositories. Higher tiers can also offer features that are otherwise locked, and better support from the provider. With a tiered approach the provider gains a steady income with, for example, monthly subscriptions and often good customer commitment but also enough flexibility for the customer. The contents of each tier have to be planned carefully so that the customer has a reason to upgrade to the next tier when their business grows. [10]

The most flexible approach is usage based pricing. It is especially attractive for small

businesses that do not want to make large investments or cannot easily predict their rate of growth. From the provider's point of view, the downside of this model is that customers can easily move to a different provider and predicting revenue becomes hard. [10]

Freemium models are pricing models that have a lot in common with perpetual, tiered, and usage based licenses, but offer a free trial or tier. The model can be based on time, free capacity up to a certain level, or a stripped set of features. The service can also be free to use, for example, for educational, non-profit or research work. The idea behind a freemium model is to convince the customers that the service is what they need. Free service levels have to be good enough to convince the users that they need the service also in the future when their business grows. The balance between the free tier and the first premium tier is critical to the success of the service and so is selecting the correct criteria on which the model is based on. If the customer is a large enterprise, the end users may not be the ones selecting their tools due to company policies. For these kinds of situations a free trial may be the best option. For a small company, a free tier offering, for example, limited features for a small amount of users can be a good way to introduce the service and as the company grows they may move on to the premium version. [10]

2.5. Advantages of Cloud Computing

In a traditional client-server model, the provider has to predict the number of users at peak times to be able to provide a good service. If there are not enough servers, the service may become unreachable to some potential customers and they may never come back thinking that the service is never good. On the other hand, if the provider has enough computing power to handle huge peaks, for example sales during Christmas holidays, many CPUs may be unused during other times, and revenue is lost for cooling and electricity. The same might happen during nights if most customers come from timezones close to the provider. These problems, called under and over provisioning, are also presented in Figure 2.1. Hosting the service in a cloud environment enables automatic scaling. This means that when the amount of users increases, new virtual machines are provisioned and the quality of service remains good. When the amount of users decreases, the virtual

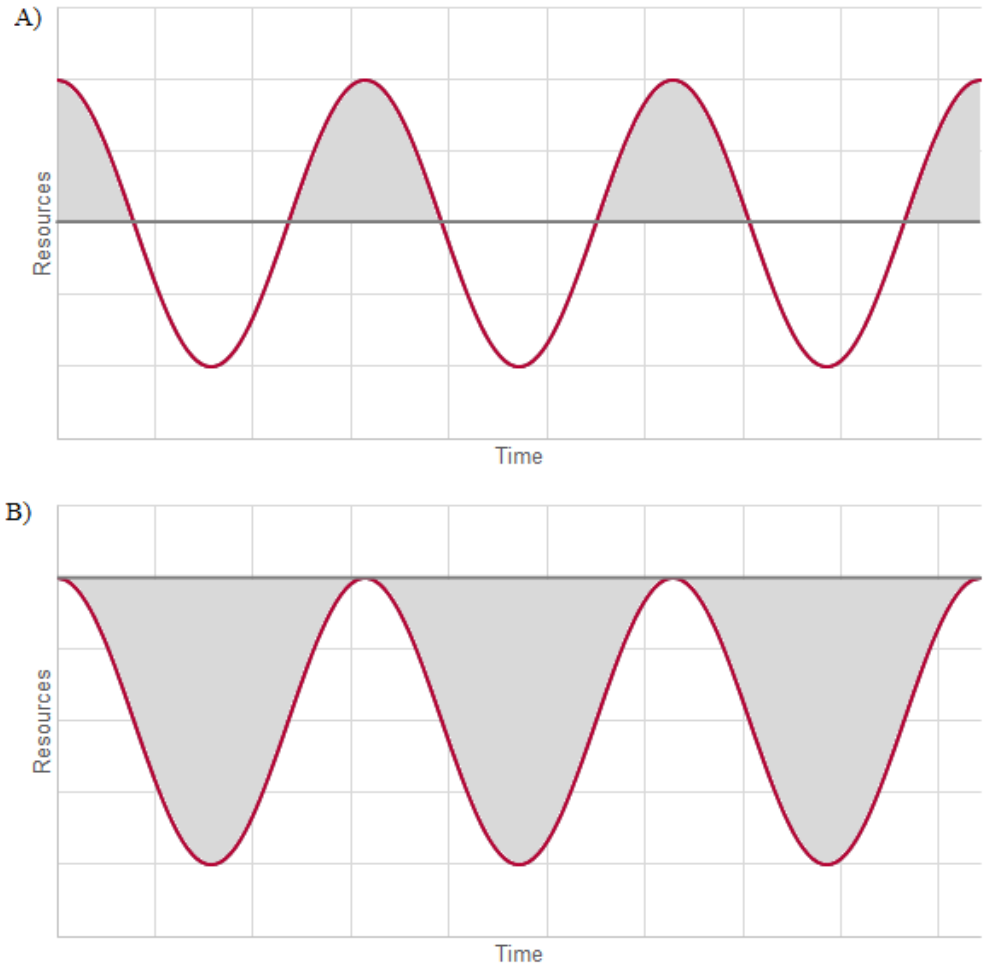

Figure 2.1: Under and over provisioning of server resources. The red line represents an imaginary amount of needed resources with peaks during daytime and lower usage at nights. The dark gray line represents the resources available. The shaded area represents A) the amount of resources that would be needed to fulfill the need and B) the amount of extra resources that increase expenses but are not needed at all times. [2]

machines are shut down. The service provider pays the cloud provider only for usage, so unnecessary expenses can be avoided. [2]

High availability can be achieved in a cloud environment if the application is able to balance the traffic between instances. In an active-active setup, multiple instances of services are running behind a load balancer, which distributes requests between the services. If a service becomes unavailable for any reason, a new one can be started or the traffic can be routed to the ones that are still running. An active-active setup might not be necessary if the amount of users and requests is low enough. In that case an active-passive setup can be used to guarantee high-availability. The active instance of a service handles incoming

requests and the passive one is ready in case the active service fails. If the active service is not responding, the passive service is made active. A new instance is created, and it takes the role of the passive service. However, the failover is not as fast as in an active-active setup. If the application is business critical and downtime has to be kept very short, an active-active setup is better. Cloud providers have many different kinds of services that can be utilized when configuring an active-passive or active-active environment, for example, load balancing services and the ability to manage IP addresses. Some cloud providers have data centers in multiple countries around the world. Having a backup of the end users' data ensures that even if one data center is unavailable due to, for example, a blackout or some natural disaster, the service is still available through another data center. [11]

Using a software that is deployed in a cloud can be tempting for companies because they can reduce the amount of their own IT infrastructure. In addition to the benefits gained from scaling, deploying to a cloud removes the need for servers that can become expensive when there are a lot of users. Not only do they need electricity and cooling but also administrators that take care of updating the operating systems and other software running on the server. Depending on the contract and used service model, the cloud provider may be responsible for this. Cloud environments are also useful to anyone who requires computing power or hosting services for a limited time, for example, for compressing a large personal video library or renting a private game server for a couple of hours [12].

Cloud providers have to have good security since in the case of a security breach many of their existing customers might switch to another provider. Providers have security experts as employees to an extent not possible for many other companies. The structure of their environment is also often more homogenous than in traditional data centers making it easier to test and manage. [13]

2.6. Disadvantages of Cloud Computing

Cloud providers have highly varying platforms and supported technologies. With high variability, it should be easy for companies to find a suitable cloud provider but the se-

lected provider can have a big impact on portability. If the cloud user becomes dependent on the APIs or, for example, databases provided by the cloud provider, it may be difficult to move to another provider if the service turns out to be problematic.

Studies have shown that CPU performance and the time it takes to execute disk I/O operations changes depending on other users sharing the same computing resources in a virtualized environment. Bursty applications, such as web services that suddenly gain popularity and publicity in social media, can cause jitter and varying response times for other applications using the same CPU. Specifying limits for CPU usage can also lead to higher response times since some of the resources are left unused. Cloud providers can use certain scheduling algorithms to mitigate this problem. [12][14]

Virtual machine instances sharing hard drives can also impact each other's throughput negatively. Allocating certain amount of disk I/O operations for each virtual machine instance and isolating them does not, however, solve this problem. Limits can increase latency and lower throughput since the hard drives cannot optimize the I/O operations as efficiently. By configuring the application correctly, for example, by increasing a database's memory buffer so multiple write operations are done as a single run, the user can affect the performance of their application and also other applications using the same cloud environment. [15]

Cloud providers have clear specifications on their network performance, but the performance of the cloud's internal network is either not reported or is advertised to be good with any kind of virtual machine instance. Studies have shown that the size of the virtual machine instances clearly matters when the intra-cloud network performance is measured as network throughput in Microsoft Azure and Amazon EC2. The available geographical regions that were setup by the providers at different times and possibly with different technologies were reported to have significant impact on the network throughput. The throughput was also shown to be limited by the smallest instance participating in the network traffic, which means that larger instances will suffer when used with smaller instances. These differences are not transparent to users, and without measurements, the bought service may have worse performance than is possible with proper configuration. The virtual machine instances were reported to have a steady performance throughout

their lifetime in the Microsoft Azure environment. However, identical instances may have varying performance which could only be fixed by redeploying the instance. This variation between identical instances was not detected in Amazon EC2. [16][17]

Distributing data is good for crash recovery, but it also has its downsides. Laws concerning the storage of personal data vary between countries. Personal data is the data that can be understood to concern a person or household. For example, Finnish personal data can only be stored in Finland if the person has given permission, it is necessary for an organization, or the person is a customer of the company handling the data. Sensitive data is the personal data describing a person's ethnicity, political ideas, religion, criminal or health records, sexuality, and social security related benefits. It can only be stored if the person has granted permission or if the data is needed by a related organization, such as a health care facility. Personal and sensitive data may be moved to another country in the European Union (EU) or European Economic Area (EEA). They can only be moved outside these countries if the European Comission has agreed that the country fulfills the requirements concerning data security. [18][19]

The Safe Harbour Privacy Principles were used to breach the gap between data privacy laws in Europe and the United States. Personal data was allowed to be transferred to a U.S. company if they followed the set of principles. Many of the largest cloud providers are American companies, and recently the Court of Justice of the European Union ruled that the Safe Harbor Decision is invalid [20]. The reason was that cloud providers that are American or have a branch in the United States cannot guarantee that data is not accessed by a third party. These companies have to comply with U.S. laws, which grant law enforcement authorities and intelligence agencies right to request data from the cloud providers. Even though the company follows U.S. laws, non-U.S. citizens outside the U.S. are not protected by the United States Constitution, giving agencies lots of power. If a company stored data in a cloud service operating only in Europe, the data would be protected by European laws, including the EU Charter of Fundamental rights which affects also people outside Europe. Personal data can be stored outside Europe only if the contracts clearly specify the required level of data security required by the European Comission or the data remains inside a corporation following the European laws in all

its branches. Negotiations between EU and U.S. in early 2016 have lead to the creation of EU-U.S. Privacy Shield, the aim of which is to guarantee a sufficient level of privacy according to European standards. However, this agreement is not yet in use. [21][22]

Even though the security of a cloud environment is well tested and managed by professionals, it is also subject to more attacks, since cloud environments contain a lot of data that is of interest to criminals. The data may have previously been stored behind a company's own firewalls, but in the cloud the environment is accessible from the Internet. Even the administrative interfaces can be exposed, offering attackers lucrative attack surfaces. In addition to the deployed applications, storage, and systems taking care of virtualization, there are also many other systems that can be exploited. Attacks against systems taking care of resource monitoring, usage monitoring, automated scaling, and data replication could lead to data loss and decreased availability. Attacks can also come from inside the cloud. Someone using the same logically separated physical resources could exploit a vulnerability and gain access to all the data stored on the physical device. Even though the cloud itself would be secure, proper application security is crucial. It is always the responsibility of the cloud user. In an IaaS environment, also the deployed virtual machine is the responsibility of the cloud user. [13]

3. ASSESSING CLOUD COMPATIBILITY

Cloud environments have reached the level of popularity where companies use the term cloud for marketing purposes as soon as they have something running in cloud, but not all software can be moved to a cloud. In some cases the benefits of a cloud environment are not big enough to justify the required work. The business perspective is discussed in Section 3.1. In Section 3.2. we take a look at technical aspects that make moving the software to a cloud environment easier. Technical issues that can prevent the software from being deployed to a public cloud are discussed in Section 3.3. In Section 3.4. we take a look at design patterns that are especially suitable to a cloud environment. Finally, a lightweight method for assessing cloud readiness is presented in Section 3.5.

3.1. Business Perspective in Assessing Cloud Feasibility

Moving to a cloud does not only affect the application itself. Instead, it affects the whole business. The changes in costs and revenue are discussed in Subsection 3.1.1. Ways to ensure the privacy and security of data are discussed in Subsection 3.1.2. Some third party software licenses are such that they cannot be used or will be expensive in a cloud environment. Issues with licenses are discussed in Subsection 3.1.3. Problems caused by vague service level agreements (SLA) are presented in Subsection 3.1.4.

3.1.1. Economical Changes

A cloud application can offer a smoother experience to the end customer than an application deployed on traditional servers, but companies have to carefully consider the economical changes that come with cloud environments. If an application is deployed

in-house, the costs caused by hardware, cooling, electricity, and salary for administrators should be compared with the different payment options provided by cloud providers. If the costs of a cloud deployment seem to be higher than the current setup, it can still be a feasible option if the servers are over or under provisioned or better availability is required. Outsourcing the infrastructure also means that there will be no future costs for replacing broken or old hardware.

Data storage and networking can become problematic and expensive when the application is deployed to a public cloud. Cloud providers usually specify costs based on how much storage space is provided and how much data is transferred in or out of the cloud environment. The amounts of data transferred in a production environment has to be measured before accurate cost calculations can be done. If the application uses large datasets, a hybrid cloud can help reduce the amount of data transferred since processing can be done in an on-premises private cloud.

If the application cannot be deployed to a public cloud, a private cloud can be used instead. The infrastructure costs can remain the same but employees may require training in order to be able to maintain the cloud environment. If the administration is bought as a service from the cloud provider, the expenses will be higher and have to be weighted carefully against the benefits.

Software licenses have to be weighted carefully. Perpetual licenses are not optimal for cloud environments, but they may be a potential option for large enterprise solutions that are deployed to a private cloud. In this case a freemium model may still be a better option to convince the customer of the benefits gained from the software. Changing from a perpetual license model to any other more flexible license will have an impact on income and may even require changes in how customers' requirements can be fulfilled. With a perpetual license, it is easy to agree on creating additional features either before delivering the product or in the near future as an update, since the purchase creates the budget that is used for developing the new features. With subscriptions or a usage based pricing model, there is no clear solution for handling customers' wishes since they are buying what already exists, and the payments trickle in, for example, monthly. One solution is to offer support services, that can be used to sell new features if a customer requires a

feature immediately.

3.1.2. Data Security

Storing data on hardware managed by a third party requires some consideration especially if the data is critical for the business or contains personal information of customers. In order to follow data security laws, many cloud providers offer the option to store data to a data center located in a certain region. In addition, the customer should make sure all the backups are also stored in a suitable region. For an organization it may be possible to negotiate whose laws are followed in different situations. Service level agreements (SLA) should specify who has access to the customer's data and environments, and also clearly state who owns the stored data. It is also recommended to find out how the provider follows standards and whether they check the background of their staff. [19]

Removing a file does not mean that the file is completely gone. Instead, the hard drive location is free for writing. The SLA should specify if the data has to be overwritten so that it is no longer accessible after deletion. It should also take into account special cases, such as when the service is terminated or something unexpected happens. [19]

Current legislation in Europe and the U.S. does not protect data from American agencies such as the National Security Agency (NSA). Even though European data is stored physically in Europe on servers owned by an American company, American agencies can access the data. In order to fully protect the privacy of customers or employees, or protect company secrets, the data has to be encrypted. However, using encryption services provided by a cloud provider does not protect it from outside access, if the encryption keys are also handled by the provider.

Very sensitive data should be secured in all its states: while at rest, in transit and in use. Communication standards support encrypting data in transit but the other steps are not as simple. Encrypting data in a public cloud might not be possible if the data is stored in a multi-tenant database. In a private database, encryption can still have a big impact on performance. Encrypting data in use is an even harder area which includes encrypting the whole memory. [13]

It is important to carefully consider all options when selecting a virtual machine image to be used in an IaaS environment. Publicly available images may have already been scanned for vulnerabilities by people with malicious intent. An image could also be created so that it has back doors. It will take time and effort to create a secure virtual machine image and it also has to be kept up to date. [13]

3.1.3. Third Party Software Licenses

Licenses of bought third party software can be problematic in a cloud environment. Some software has to be licensed based on the number of users. Using such software as part of a system running in a cloud may not be possible since it is impossible to know the number of different users in advance. There may be an option to use the number of CPU cores as the basis for the amount of needed licenses. This, however, comes with the problem that the license should clearly state whether it means physical or virtual cores.

Oracle's databases are widely used and have complicated license agreements. In many cases they require a license for each physical CPU core in the system running the Oracle product. Oracle has named Amazon and Microsoft as Approved Vendors and has special licensing requirements for their cloud environments, but many of the popular virtualization technologies, for example from one of the largest providers VMware, are not approved. As a simplified example, based on current policies, when creating a private cloud managed by products from VMware and running Oracle software in a virtual machine on just one of the physical servers, the number of required licenses is equal to the number of physical CPU cores in the whole system times a factor determined by the CPU model. This is a good example of a possible scenario, but does not apply to all Oracle products since licensing is slightly different for every edition of, for example, Oracle Database. These kinds of issues can have a huge impact on what kind of environment and what third party software, such as databases, can be used. The whole technology stack and licenses have to be carefully studied in order to keep costs reasonable and to avoid license infringement. [23][24]

3.1.4. Service Level Agreements

Many cloud providers have promises of at least 99.95% availability, meaning that their services are unavailable only 0.05% of the time. If the time is calculated monthly, that is 22 minutes. The SLAs, however, contain more specific details about what is considered downtime. The Amazon EC2 SLA defines unavailability as an instance not having outside connectivity. Amazon's Regions consist of Availability Zones and the SLA requires more than one unavailable Availability Zone in the Region for the customer to be entitled to compensation. Planned maintenance is not considered as unavailability. Amazon offers Service Credits as compensation and they can be used to pay for subscriptions in the future. Microsoft Azure has a SLA very similar to Amazon EC2. Neither of the SLAs mention, for example, data ownership, data access or performance, meaning that customers are only entitled to compensation during total unavailability. It is also the responsibility of the customer to submit a claim and provide exact times of the outage and logs as proof. These SLAs can be enough for small businesses, but for business critical software possibly handling sensitive data, the SLAs should be negotiated to contain clearer rules about data and possibly better compensation for unavailability. Downtime can be expensive and receiving compensation in the form of service discounts may not compensate nearly enough. However, the actual measured uptime of cloud environments has been increasing and some providers have managed to provide an uptime higher than 99.95%. [4][6]

3.2. Beneficial Design Patterns

In this section we take a look at aspects that are part of good software design and development and especially important for cloud applications. Decoupling, statelessness, and automation of delivery are discussed in the following subsections.

3.2.1. Decoupling of Components

Coupling of components, such as classes or applications, means the level at which components are communicating with each other. Tightly coupled components can, for example,

affect the internal state of each other or use common data. A change in one of the communicating components or data usually requires a change also in the other component. Loose coupling can be achieved by having components that offer each other services with a clearly defined input and output. The actual implementation of the service can change without affecting other components as long as the interfaces remain the same. A service or database schema can also be completely replaced with another one. When there is an interface which handles providing the requested data, the components can be sure that they will receive the data in the same format as before.

When the components of an application are tightly coupled, new versions of the software have to be deployed as a whole. Otherwise communication between components may not work. Loose coupling enables deployment of new versions of a single component if the interfaces have not changed. With backwards compatibility and versioning of components, the latest versions can provide the service to older components.

3.2.2. Stateless Applications

An application's state means the data that is available at a certain point of time during the execution of the application. A stateful application stores data in objects whereas a stateless does not maintain the data across multiple calls. Each call to the application is handled individually and must contain all the necessary information for the operation. A stateless application is more scalable than a stateful application, since any instance of the application can handle any of the calls. If the state is persisted across multiple calls, the calls would have to be done to the same instance. Existing data also prevents multiple users from using the same objects.

Statelessness increases the scalability of an application. In a cloud environment, a stateless application can be scaled up and down by adding new instances or by shutting down existing ones. An instance can be shut down without checking whether on-going long term interaction exists between a user and the instance, since there is no risk of losing data. New instances can be created and the next requests that the client sends can be forwarded to any of the instances. A stateless application is also more resilient to failures,

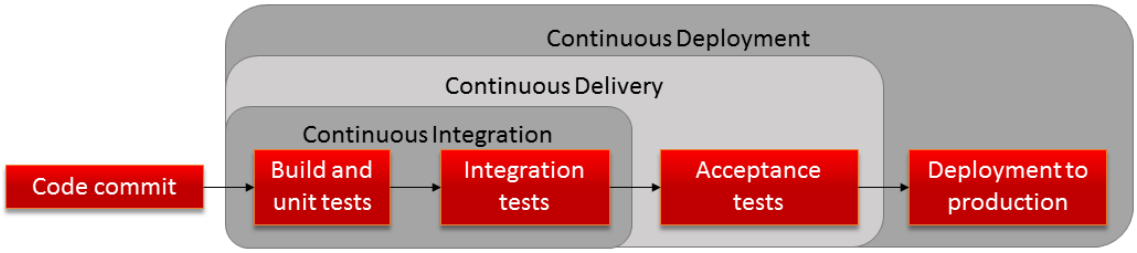

Figure 3.1: The steps that are automated in continuous integration, delivery and deployment are shown in each of the frames.

because other instances will be able to continue the execution. [25]

3.2.3. Automated Delivery

Business critical software running in the customer premises has to be stable and changes must be infrequent. When this type of software is moved to a cloud environment, large changes may be required in order to improve scalability and a very different environment may bring unexpected problems. Automated testing is vital in order to maintain stability but the release cycle should also be quick in order to deploy fixes to new kinds of problems that may arise from dynamically scaling components. Continuous integration, delivery, and deployment are different automation levels of testing and delivering releases. The level of automation is depicted in Figure 3.1.

Continuous integration is the automation of building the software, running unit tests, deploying to a test environment, and running integration tests. The cycle is started after every push to the version control system in order to make sure that the application is functional. This level of automated testing does not guarantee that the built software is ready for a release. The term continuous integration comes from developers continuously integrating their changes to the source code repository. [26]

Continuous delivery takes automation one step further by automating acceptance and performance tests in an environment similar to production. These tests can be triggered, for example, nightly and a package that passes the tests can be considered something that could be released. The decision to release is done manually and releases can be done later in order to have more content in the package and not to disturb the production environ-

ment so frequently. The actual deployment can be automated after the decision to release has been made. [27]

The only difference between continuous delivery and continuous deployment is that the latter automates everything from pushing a change set to a version control system to deployment. The software can be deployed to production quickly, and if a problem occurs, the latest change is probably the reason for it. Since the change is small compared to what would be deployed if the release was done once every two months, it is easy to go through the changes and see what could have caused the problem. This is how continuous deployment can help reduce downtime. [27]

Blue-green deployment is a technique that can be utilized in continuous delivery or deployment. Blue and green are names for two environments. One is in production and the other is a new version that is going to be deployed to production. If the blue is the current production system, green is deployed, for example, as another virtual machine to the same cloud environment. The green environment undergoes the final stages of testing, and once it is considered ready, the load balancer can start forwarding requests to it instead of the blue one. The blue environment can be left running as a backup, and if there are problems with the new production environment, it is easy to revert to the previous version. Once the green one is considered to be stable, the blue one can be used as a test environment for the next version or deleted. A similar technique to blue-green deployment is canary release. The only difference is, that instead of switching from one environment to the other in one go, the new version is installed on some of the servers and they handle part of the load. The load of the new version increases slowly as more instances of it are taken in to use. This way possible problems are detected before all end users are affected by them. [27][28]

3.3. Possible Hindrances

Cloud deployment requires careful planning of many aspects of the whole system. Public clouds are not especially friendly for applications requiring high performance. This issue is discussed in Subsection 3.3.1. Dependency on the services provided by a cloud provider

can lead to vendor lock-in which is discussed in Subsection 3.3.2. In Subsection 3.3.3. we will take a look at integrations with other systems that need to communicate over the public Internet in order to reach the cloud application.

3.3.1. Heavy Disk and CPU Usage

Virtual machines sharing a hard drive will affect each others' performance. If an application relies heavily on database operations, a cloud environment can turn out to be problematic. With proper configuration and minimizing disk I/O operations, latency can be reduced but response times may still be lower than by running the database on a dedicated server. Similar issues have been detected when running CPU intensive applications. Small tasks that can be distributed to many instances with a load balancer can most effectively utilize the benefits of a cloud environment. For applications that perform a lot of disk I/O operations, one option may be a hybrid cloud deployment. The parts of the system that require a lot of resources can be deployed to a private cloud in order to minimize the negative impact caused by other application sharing the hardware resources. Another option is to rent dedicated instances that are running on hardware dedicated to only one customer. For example, Amazon offers dedicated instances but the price is higher than for normal shared resources.

Due to the lack of transparency in intra-cloud network performance, the traffic between rented instances should be measured, especially in the case of performance sensitive applications. All parts of the system should be deployed on similar instances to avoid losing some of the network performance of the larger instances. Published results of performed studies and comparison of different providers can give valuable information on which cloud environment is optimal for a certain use case and what kind of issues should be taken into account.

3.3.2. Dependency on the Service Provider

In the case of PaaS, the number of available technologies may be limited. For example, Google App Engine only supports Java, Python, PHP, and Go programming languages and

certain databases [5]. A company might be developing their applications in languages not supported by a PaaS provider and significant work would be required in order to move that environment. Using an API provided by the platform can also result in vendor lock-in if no other provider has similar APIs. Moving away from that provider will be problematic since reimplementation of that part of the system is required.

Using one service provider is a single point of failure [2]. If the provider goes out of business the whole application will be unavailable until it can be moved to another environment. If the service was a platform with provider specific APIs, the migration to a new environment may prove to be a significant investment and downtime can be long. Carefully considering the supported technologies and making a backup plan is important when planning on using a PaaS. For highly customized systems, an IaaS environment is a safer option.

3.3.3. Integration with Other Systems

Communication in a complicated system running in a hybrid cloud requires careful design. Data needs to be transferred across the public Internet and has to be kept safe. Often single sign-on is required for the convenience of the users.

Applications deployed to a cloud may need to communicate with other applications running in a company's private network. A Virtual Private Network (VPN) can be used to extend the private network over the Internet. VPNs can be remote-access or site-to-site, meaning that they either connect a device using a VPN client to a network or connect two networks. Some cloud providers offer site-to-site VPN services, requiring the customer to setup a VPN gateway in their network. The benefit is that the customer can then access the cloud environment as if it was part of the company's own private network, and the communication is encrypted. If the company has centralized user management that can be used for single sign-on for multiple applications, using a VPN can be used to extend the single sign-on to work with the cloud applications. On the other hand, using a VPN adds costs and overhead since messages have to be encrypted and decrypted at the gateways. [29]

Authentication and authorization become more complex in a service oriented architecture, where each service is independent and scalable. Instead of just signing in to one application, the user has to be authorized for each service that they need. A solution is to have a service that handles authentication. After the user signs in, the service provides a token that can be sent with requests to the necessary services. JSON Web Tokens (JWT) are widely used with a service oriented architecture. [30]

For basic SaaS applications, many of these approaches do not work or are not needed. Often the user can only create an account and change the settings that are directly related to the usage of the application and cannot, for example, set up a VPN connection or configure single sign-on. In many cases this is not even necessary but single sign-on offers better usability, especially when the application is used for work.

3.4. Cloud Friendly Architectural Patterns

In this section we take a look at architectural patterns that are especially good for a cloud environment. Representational state transfer is often used for building APIs for web applications, and is discussed in Subsection 3.4.1. In Subsection 3.4.2. we take a look at service-oriented architecture and in Subsection 3.4.3. at microservices which is similar to a service-oriented approach. Finally, caching is described in Subsection 3.4.4.

3.4.1. Representational State Transfer

Representational state transfer (REST) is an architectural style developed for distributed hypermedia systems. The unit of information in REST is called a resource which is any information that can be given a simple name. A resource has a representation which is stored on the server. The representation contains data and metadata. Every resource can be identified with a Uniform Resource Identifier (URI). [31]

The communication between a client and a server is done using stateless HTTP requests in order to fetch or manipulate representations. The messages are self-contained, meaning that they contain all the necessary information for fulfilling the request. This way the state

of the clients does not have to be stored on the server. REST uses hypermedia as the engine of application state (HATEOAS), which means that the server informs the client what actions are available for a requested resource and what URI to use for each of the possible actions. A RESTful application can have very low coupling between components, and therefore, separate components can be deployed independently and compatibility between different versions is easier to maintain. [31]

3.4.2. Service-oriented Architecture

Service-oriented architecture (SOA) is an architecture style which aims to reuse existing code and lower the amount of dependencies between software components. Every component has a clearly defined function that other services or consumers can use through simple interfaces. Since its the service's goal to solve a consumer's problem, communication is done using descriptive messages that do not affect how the system behaves. A schema is used to restrict the structure of messages, but in order to fulfill the goal of reusability, the services have to be extendable. That is why a good balance needs to be found between restriction and extendability. SOA does not specify how consumers and services communicate. Web services handle the communication using internet protocols such as HTTP, FTP, and SMTP. Design patterns that can be used for communication between web services are Simple Object Access Protocol (SOAP) and REST. [32]

3.4.3. Microservices Architecture

Microservices architecture is similar to SOA in the sense that components are services that communicate with each other using internet protocols. Components can be written in different programming languages and they offer an interface for other services or consumers. Microservices have raised much debate on whether it is just a new name for SOA, but Lewis and Fowler [33] claim that it is an architectural style that more people should consider because of its benefits. In addition, SOA, SOAP and web services are easily and mistakenly thought to always belong together. The goal of microservices is to remind that monolithic enterprise applications can be broken down to services that scale better, in

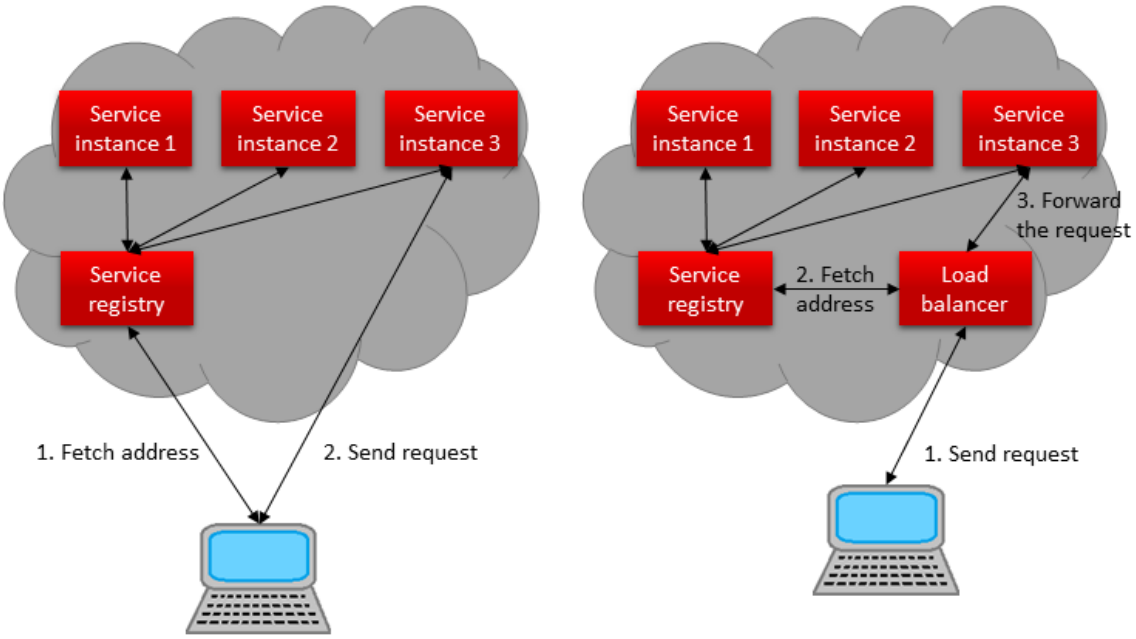

Figure 3.2: In client-side discovery (left) the client knows the address of the service registry and handles load balancing itself. In server-side discovery (right) a separate load balancer is used.

other words an internal SOA, where as web services aim for independent services that consumers can find using registries and which can be used to build larger systems. [33]

A monolithic application can only be scaled horizontally behind a load balancer and, such a system is usually deployed as one large application. A change in one small part, for example a class, in the monolithic application requires redeploying the whole application. Independent services can be deployed separately and scaled non-uniformly. There might be multiple instances of service A on multiple hosts but only one instance of service B on one of the hosts. The monolithic application does not have to be completely split into services and, for example, having a monolithic core with just the bottlenecks as highly scalable services is a feasible approach. [33]

Microservices that can start or stop based on the load have dynamically changing addresses so a client cannot know where to send a request. Service discovery mechanisms, presented in Figure 3.2, can be used to store the location of available services and monitor their health. A service can register itself to a service registry and continue sending heartbeat throughout its lifetime, or a separate service manager can be used to handle starting and stopping services. The service manager also informs the service registry what the

address of each available service instance is. If client-side discovery is used, the client is aware of the registry's location and queries the addresses of service instances. The client is also responsible for handling load balancing. In comparison to server-side discovery where a separate load balancer queries the registry and forwards the client's requests, client-side discovery adds complexity to clients but there are less parts on the server. [34]

3.4.4. Caching

While load balancers and application instances can be scaled without huge problems, databases are not as simple. Adding more database instances requires duplicating or redistributing the stored data, which can be time consuming. Distributing the data to different instances makes reading and writing more complicated. One way to increase the performance is to reduce the amount of database operations by caching. Using caches can improve performance especially for read intensive applications. Instead of always reading data from the database, the most frequently used data can be stored in memory or on the disk of a virtual machine instance dedicated to caching. How to store the cached data depends highly on the size of the cache, the size of a data unit, number of requests, and other requirements such as persistence. The services and infrastructure provided by different cloud providers are different, and the results of research done on the cost effectiveness of caching in an Amazon environment cannot be generalized to other environments. In general, caching in memory is fast but tends to be more expensive. In a tiered architecture, a caching tier could be added between the application and database tiers. When data is requested, the cache is checked first, and if the data is not available, it is fetched from the database. Another option is to have a caching layer between the load balancers and application instances. [35][36]

3.5. Cloud Readiness Assessment Method

This section describes a method that offers help when a company is interested in running their products in a cloud environment. The aim of this method is to be lightweight and to show the main problems that the company would have to solve before migrating.

Three flow diagrams presented in Figures 3.3, 3.4, and 3.5 contain questions that can be answered with yes or no. The questionnaires are used to analyze the need for a cloud environment, the overall feasibility of using a cloud environment, and the cloud friendliness of the current software architecture.

The first questionnaire presented in Figure 3.3 can be used to analyze whether there is an actual problem that could be fixed by moving a product to cloud. The main advice is that if the application is facing performance issues, public clouds are a good way to add resources and especially to handle spiking and variable loads. They are also good for preparing for growth. Security can be better in public clouds where the provider is responsible for the underlying infrastructure, since providers often employ security experts. When an application is deployed to a cloud environment, it should be monitored and automatic restarts should be configured. This helps reducing long downtimes. Clouds can also be used to save on infrastructure costs. In the case of a standalone desktop application, migrating to a cloud can offer better accessibility since users can access their data with any device that can connect to the application. Cloud applications tend to use subscription or usage based licensing, which can also be useful for luring in potential customers. If there is a need to start offering a desktop application as a web application, a cloud environment is a good choice if the company does not have existing hardware resources.

The second questionnaire presented in Figure 3.4 maps the overall cloud compatibility of the application and offers suggestions on what kind of deployment models are suitable. Applications with sensitive data can be deployed to public or private clouds hosted by a third party, but security and data ownership has to be clearly defined in the SLA. Private clouds running on the data owner's own infrastructure are also an option, especially if the application transfers large amounts of data since data storage and network traffic can become expensive in public clouds. Private clouds provided by a third party are an option for applications that perform a lot of disk I/O operations, because they are separated from interference caused by other users. Multitenancy may not be feasible for applications that are highly customizable for each customer. A separate environment may be needed for each

Figure 3.3: Questionnaire for analyzing the need for a cloud environment. The questionnaire starts from the ellipse and ends in one of the colored nodes. A green node represents a clear need for a cloud environment, a yellow node means that there are potential gains, and a red one that migration to a cloud might not bring benefits.

Figure 3.4: Questionnaire for analyzing the overall cloud feasibility of the software. The questionnaire starts from the ellipse and ends in one of the colored nodes. A green node means that the application is suitable to a public cloud and a yellow one that a public cloud is an option but there are some concerns. A red node depicts that public clouds are not very suitable. In this diagram, private cloud refers to a cloud environment deployed to a private network. The term 3rd party cloud is used to refer to public and private clouds hosted in the providers network.

of them. Locally hosted private clouds can also be a good choice in situations where a large system is communicating with many other applications running in the private network. Another option is to use a VPN.

The third questionnaire presented in Figure 3.5 maps potential problems in the current architecture of the application. Stateless applications can serve many users because they

do not keep track of every user. State would hinder the application's ability to scale. Loose coupling is also good for scalability. Monitoring is crucial in order to achieve high availability and scalability, because the state of the service can be used to start up new instances when load increases or some instances are down. This also applies to an application that is just a single lightweight instance. A monolithic application may require changes in order to gain these benefits. The application's bottlenecks could be separated into scalable components. Continuous delivery helps with delivering fixes and new features fast and without manual work.

Figure 3.5: Questionnaire for analyzing the cloud readiness of software architecture. The questionnaire starts from the ellipse and ends in one of the colored nodes. A green node means that the application architecture is good for a cloud environment and a yellow one that some issues should be addressed. A red node depicts that clouds are not very suitable for the existing architecture.

4. CASE STUDY

This chapter presents OSS/BSS systems and the architecture of Comptel InstantLink, a provisioning software product. In Section 4.1. we take a look at OSS/BSS systems and Comptel InstantLink. Software defined networking (SDN) and network functions virtualization (NFV) are new technologies that drive communications service providers (CSPs) towards cloud environments, and we take a look at these technologies in Section 4.2. The results of using the cloud compatibility method in evaluating Comptel InstantLink are presented in Section 4.3.

4.1. Provisioning Software

Operations support systems, together with business support systems, (OSS/BSS) are used by operators to manage everything related to their network and business. The OSS is used for the operator's back-office work, which includes communicating with the network elements (NE) in order to create users and services, also known as provisioning, and activating services. The service inventory is also a part of OSS, and it is used to store data about available services. The BSS is used for handling work that is directly related to customers, such as billing, orders and storing information about customers. These two parts are closely related since their components are constantly exchanging information and some can be considered to be part of both OSS and BSS. [37]

Customers can use their services after all the subscribers and services defined in the OSS/BSS are configured in the network elements. Provisioning software, such as Comptel InstantLink, which is shown in Figure 4.1, is capable of handling the communication between OSS/BSS and the network. Comptel InstantLink consists of a Request Processing Engine (RPE), one or more Service Module Engines (SME), one Task Engine, and a Net-

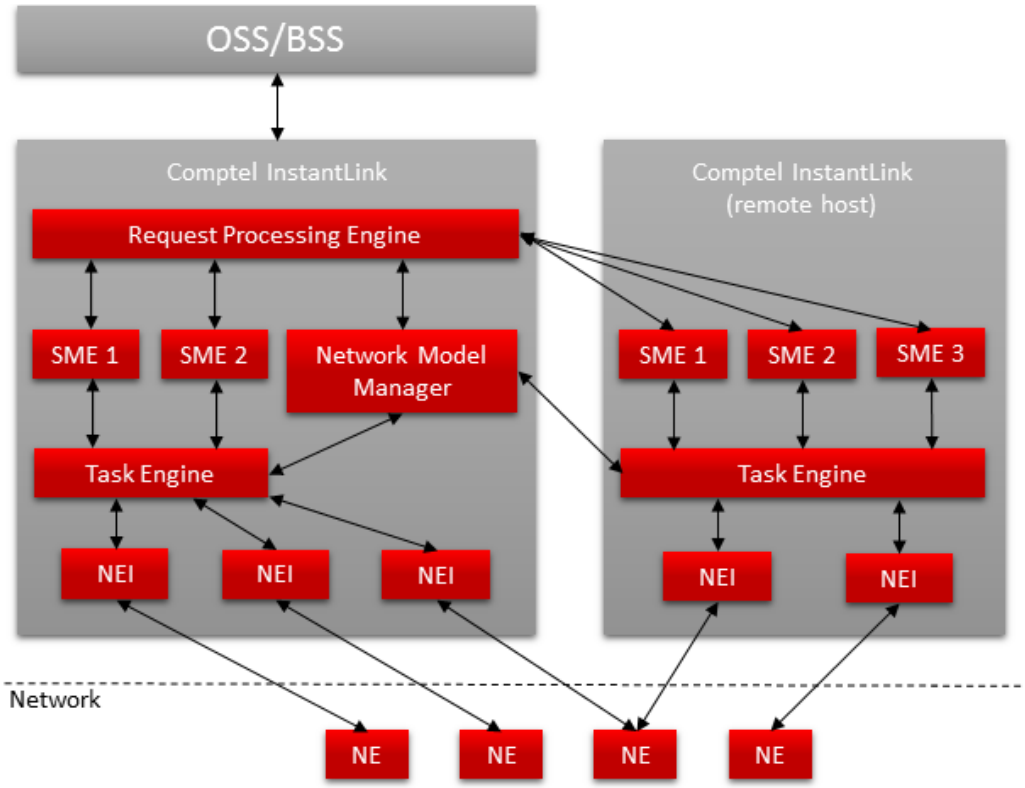

Figure 4.1: Comptel InstantLink installed on two hosts. The main host is running the RPE, two SMEs, a Task Engine, and a Network Model Manager. The secondary host has three SMEs and a Task Engine. The Task Engines communicate with network elements (NE) through network element interfaces (NEI). [37]

work Model Manager. Comptel InstantLink is scalable and can be installed on multiple distributed hosts that run SMEs and a Task Engine. Communication with other products and OSS/BSS is handled using SOAP. [37]

Comptel InstantLink is a part of the software suite called Comptel Provisioning and Activation, which is installed on UNIX based servers that are usually part of the CSP's own infrastructure. Delivering the system is a project that consists of creating required logics for handling requests in SMEs, configuring settings for all the parts of the system, and installing it. The delivery projects range from small projects of just a couple of people to larger projects with dozens of people. The durations also range from months to even years, depending on how much customization needs to be done and possible delays in acquiring required hardware. [37][38]

OSS/BSS software is critical for the business of CSPs. It is the core that automates all the processes related to handling subscribers, their data, billing, and modifying or creating

services. Comptel InstantLink has to be able to handle up to a million requests per hour, and high-availability is also very important. Because the customers of CSPs have very different needs for their services in different countries, the software has to be flexible.

The RPE is responsible for handling requests and sending responses to OSS/BSS. The received requests are converted to an internal format and distributed according to priority to SMEs located on either the main host or the remote hosts. If the load is too high or the system is shut down, the RPE stores unhandled requests to a relational database and continues sending them when it is able. In addition, the RPE locks requests when multiple requests would affect the same subscriber, in order to ensure that the operations are performed in the order in which the OSS/BSS sent them. The requests are also handled in priority order using an execution queue. The lock list and the execution queue are stored in memory by the RPE process. All other components register to the RPE which maintains information on their status. [37]

SMEs contain an interface for plug-in modules that can be used to configure the way requests are handled. A request can be simply routed as a task to a network element or can be handled with logics that can alter request parameters and break the request to multiple tasks. The Task Engine informs SMEs which network elements are available. If a task cannot be sent, the SME stores the task to the database. [37]

The network model maintains information of the network status and topology. It is responsible for informing the Task Engine of changes, for example, when a NE is unavailable. If there are multiple hosts, the Network Model Manager decides which Task Engines handle which NEs. [37]

Comptel InstantLink uses a single database for storing requests and information about the system configuration and network elements. In a distributed setup, there is still only one database. In addition, a separate archive database can be used to store old requests. [37]

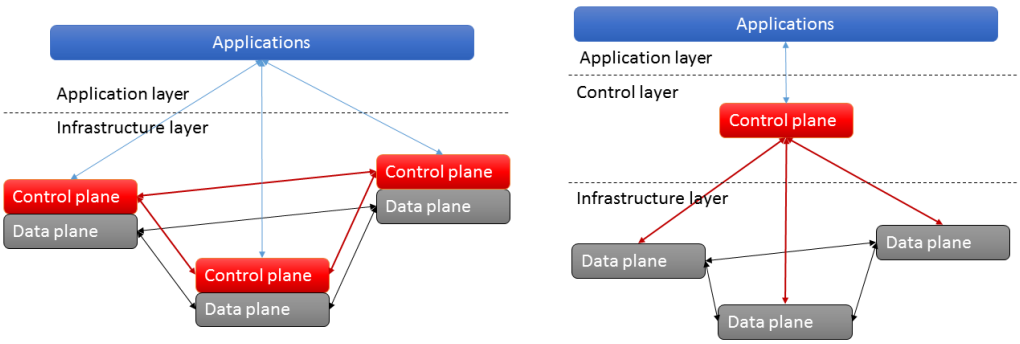

Figure 4.2: Traditional network devices (on the left) have a data plane and a control plane that decides how traffic is forwarded. SDN separates the control plane (on the right) which makes the network easier and faster to configure.

4.2. Software-defined Networking and Network Functions Virtualization

Software-defined networking (SDN) and network functions virtualization (NFV) are changing the way networks are built. In the future networks will be more dynamic and OSS/BSS systems must be able to handle dynamically changing networks. These systems should also be scalable because the number of incoming request may be more unpredictable. [39]

A network requires many different kinds of specialized devices. They can be forwarding devices, such as routers and switches, or middleboxes, such as firewalls. The devices have a data plane and a controlling plane, which decides how incoming packets are handled and it can be configured by administrators. Reconfiguring a network or updating protocols is hard and requires access to many of the devices handling the traffic. This has made Internet's infrastructure rigid and evolution slow, and, for example, hindered the deployment of IPv6. [40]

The Open Networking Foundation was founded in 2011 by network operators and service providers to standardize SDN and the OpenFlow protocol. In SDN, the data and control planes of network devices are separated, and a shared SDN controller handles the decision making of all the network devices. The separation of data and control planes is shown in Figure 4.2. SDN has clear benefits. A centralized control simplifies the network and algorithms required for forwarding. The specialized network hardware can be changed

to simpler devices and some of the middleboxes in a network can even be removed. For example, firewalls and load balancers can be implemented in the SDN controller. In SDN terminology, forwarding devices are called switches which can be categorized as pure or hybrid. A pure switch does not have on-board control and all forwarding decisions are made by the SDN controller. Hybrid switches have their own control plane in addition to the SDN controller. A hybrid switch is capable of handling simple forwarding by itself and is also more resilient to problems if the SDN controller breaks down. [40]

In addition to having hybrid switches, the control plane can be distributed in many other ways. There can be multiple physical controllers with a centralized logic. Applications accessing the network have a simplified view of it, and the switches can communicate with the local controllers. This reduces load on the controllers, but applications might not always get a correct view of the network since the network state might not be up to date in all of the controllers. The OpenFlow protocol allows a switch to communicate with multiple controllers so a controller is not a single point of failure. A control plane can also be a hybrid between centralized and distributed. A global controller can be used for decisions that require knowledge of the whole network, and local controllers, for example in a hybrid switch, can handle some traffic themselves. [40]

A small company might have their servers and client PCs communicating from their private network to the public network through a switch, a router, and a firewall. This already contains three types of specialized devices, and as the size of the company grows, there will be need for more devices for each location. A device can also be a single point of failure, which will need replacing if it breaks down. The goal of NFV is to virtualize network devices and run their functions on commodity hardware in a cloud environment. For example, the traffic from the company could be routed to a virtualized network function (VNF) that takes care of the responsibilities of the firewall and is running in the cloud. SDN and NFV are separate technologies but are often discussed together. SDN can be used to make the management of a dynamic virtualized network easier since the SDN controllers are aware of the state and structure of the network. Otherwise, when switches are removed from or added to the network, forwarding rules have to be updated in all of

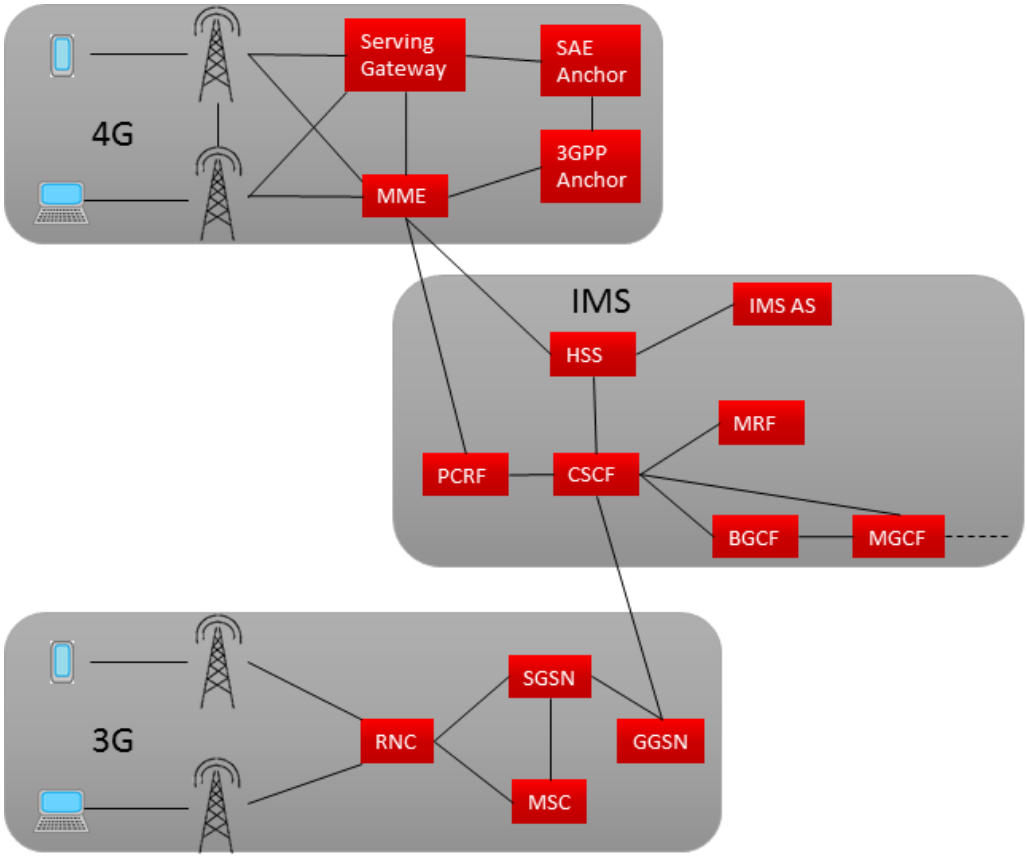

Figure 4.3: A simplified example of a mobile network. Each text box represents a network function and some of them consist of multiple hardware devices. [41]

the virtual devices. This would be laborious especially when devices can be provisioned or shut down depending on the amount of users. [42]

The networks run by CSPs are capable of handling text messages, data from different kinds of devices, and phone calls from landlines and mobile phones. Mobile networks need to be able to handle the different generations of technologies, such as 2G, 3G, and 4G. All these networks are different and require specialized hardware. A simplified example of a mobile network providing 3G and 4G services is depicted in Figure 4.3. Replacing some of the equipment with VNFs would reduce hardware costs, make it easy to scale services based on usage and generate new services. OSS/BSS systems running in cloud environments would be better prepared to scale with the dynamically changing network.

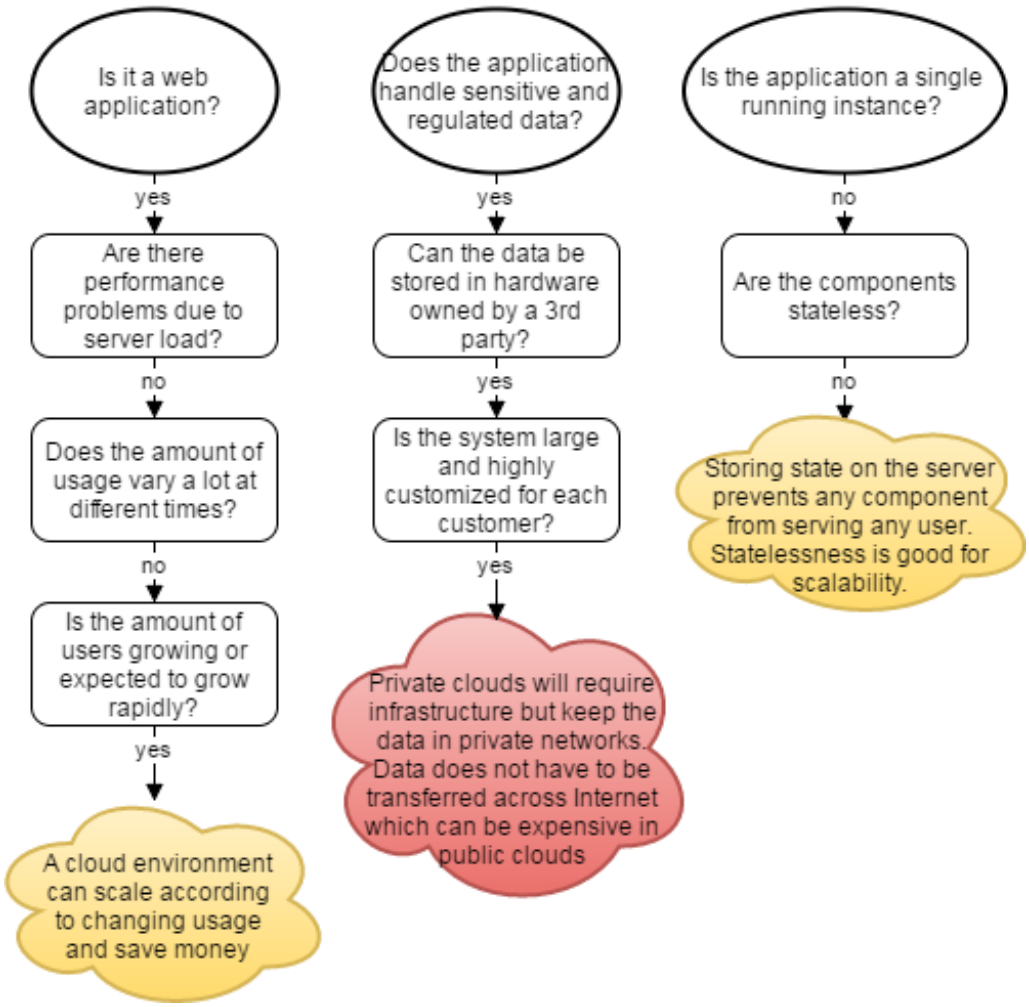

Figure 4.4: The paths that were the result of using the questionnaires to analyze Comptel InstantLink.

4.3. Cloud Readiness Assessment

Comptel InstantLink was analyzed using the diagrams presented in Figures 3.3, 3.4, and 3.5. The result of the analysis is presented in Figure 4.4. Comptel InstantLink is a web application that is deployed on customer premises as a part of a larger system. It cannot be said that every customers would have performance problems caused by the lack of server resources. In the future, the amount of data and the number of users are likely to grow, and Comptel InstantLink should be prepared for this. According to the diagram, a cloud environment could be used to prepare for this growth. The data handled by Comptel InstantLink contains personal information and some CSP's have customers who require very tight security, for example, the military. In some cases the data could be stored on hard-

ware owned by a third party. Comptel InstantLink is highly customized for each customer and the diagram suggests using a private cloud deployed to customer premises. Comptel InstantLink consists of multiple components but the RPE is responsible for maintaining the locking lists and, therefore, is not stateless.

5. EVALUATION

In this chapter, we will evaluate the results obtained with the cloud readiness framework. A closer inspection on Comptel InstantLink is presented in Section 5.1. The used method is evaluated in Section 5.2. Future work on improving the method and cloud compatibility of Comptel InstantLink is presented in Section 5.3.

5.1. Cloud Compatibility of Comptel InstantLink

Comptel InstantLink would benefit from a cloud environment, because of the possibility to scale efficiently and automatically. The amount of data that CSPs handle is growing, and reducing the infrastructure costs will drive CSPs towards using cloud environments. Due to the performance requirements and data security, the best option would be to deploy Comptel InstantLink to the CSP's own premises, where the software is also close to the actual network elements. Many CSPs require a high throughput of requests and the largest companies require it to handle over 1 million requests every hour. The latency caused by communicating over public network would have an impact on Comptel InstantLink's performance, so the components must be kept physically close to each other. Because Comptel InstantLink is a part of a larger system providing functionality to also other products, it would also have to be close to the other parts of the system to reduce latency. Also the need to customize the software for each customer makes multitenancy difficult to implement.

In its current state, Comptel InstantLink would not gain the benefits of a cloud environment due to limited scalability. The components have to be made more independent, so that they can be scaled better in a cloud environment. Availability would be better when a component would not be a single point of failure and a new one would be fast to provision

as a replacement with proper monitoring and automatic deployment. Currently the RPE is a single point of failure and contains state in the locking list and execution queues. Oracle Database has a license that is not suited for all virtualized environments, and another database should be used instead. Currently logs are written to files on disk, but in a cloud environment, all the instances might not be using the same disks and would see different log files. The logs should also be stored on a different physical server than Comptel InstantLink is running on, in order to make sure that they are available if a component is down.

5.2. Method Evaluation

The diagrams in Figures 3.3, 3.4, and 3.5 look complex, but are easy to use. They are a good starting point, and help in finding the main areas of concern when considering migrating to a cloud. The result of each diagram is realistic when used to analyze Comptel InstantLink. Some of the questions in the diagrams are slightly difficult to interpret, for example, when the application is deployed on premise for each customer. The biggest short coming of the method, however, is the third diagram, presented in Figure 3.5, which is used to evaluate the cloud compatibility of the current software architecture. It does not capture all potential issues. In the case of Comptel InstantLink, the result was obtained after considering whether the components are stateless or not. Answering no, takes the user to a result node in the flow diagram, and the user never goes through questions about, for example, the usage of the file system or lack of monitoring.

Comptel InstantLink is always delivered as a part of a system. However, the method was only used to analyze Comptel InstantLink, and the complete software suite was not taken into account. The method could also be used on a large system built from multiple products, but further analysis is needed to find out how useful the result would be. One option is to analyze every part of the system separately, and considering the whole system afterwards.

All in all, the method is easy to use and a good starting point for evaluating cloud compatibility. It summarizes the potential problems of cloud environments presented in this

thesis, making them more transparent to the user. It is not, however, particularly suitable for complex and large enterprise systems.

5.3. Future Work

In this section we will take a look at further improvements to the method described in this thesis and at work required to move Comptel InstantLink towards a more cloud friendly architecture. Subsection 5.3.1. is about improvements for the method. Changes in the architecture and delivery process of Comptel InstantLink are presented in Subsections 5.3.2. and 5.3.3., respectively.

5.3.1. Improving the Cloud Compatibility Evaluation Method

The third diagram, presented in Figure 3.5, evaluating the software architecture could be improved by adding an arrow from each result node to the next question. This would be beneficial, since it is hard to create a suitable order for the questions. They are not dependent on each other, and many of them are equally important. The flow would change so that in case there is a potential problem, the user receives a tip from the nodes that currently are the result nodes, but continues answering through all the questions. Another option is to handle the evaluation of architecture in a completely different way, for example, using a check list.

The flow diagrams are a good starting point and they could be used to create a web service for cloud compatibility evaluation. The service could present the questions to the user in the same order as in the flow diagrams. The returning arrows in the architecture diagram would allow the user to answer all questions. The service could display a summary for each questionnaire, which would make the result easier to understand. It would also be possible to list additional information and, for example, give advice for further reading.

5.3.2. Moving Towards Microservices

In order to achieve higher availability and performance, the components of Comptel InstantLink should be more scalable. One way to achieve this is to gradually start moving

towards a microservice oriented architecture, in which the RPE, SMEs, and Task Engines are independent services. In order to do this, a service discovery mechanism should be implemented. A service registry is responsible for keeping track of running instances that send it a heartbeat, and in order to send a request, the client must first query a suitable instance from the service registry. A query to a service registry should also be made, when the request is sent to an SME, and when a task is sent to a Task Engine. Each component should be monitored in order to be able to recover from crashes. In a microservice architecture, each service instance should have its own database. Different services can even use different kinds of databases. The database should only contain information relevant to the service and otherwise information should be passed between services in messages. Refactoring the current highly optimized database implementation in Comptel InstantLink is a challenge.

Work is already underway to implement active-active availability for Comptel InstantLink. No component should be a single point of failure, which means that the RPE and Network Model Manager should also be present in all running instances of Comptel InstantLink. The RPE handles request locking so that there are no parallel requests for a single subscriber, and it also uses an execution queue to handle requests in a priority order. They both are stored in memory by the RPE process and could not be stored to, for example, a database due to high performance requirements. In future versions of Comptel InstantLink, the RPE in one of the running instances will take the role of master and function similarly as before. The slave RPEs accept requests, but after backing them up in the database, they forward the request to the master RPE for lock handling. If the master RPE crashes, one of the slaves takes the role of the master. This recovery is faster than the previously supported active-passive setup. [43]

The upcoming architecture makes it possible to have multiple active instances of Comptel InstantLink. By adding a service registry, monitoring, and changing logging, the instances could be running with dynamic scaling as virtual machines in a private cloud. However, in the future the master RPE may become a bottleneck. Further changes have to be made in order to move locking away from the RPEs, so that at least all instances of Comptel InstantLink are equal. If every instance would be able to function independently, they

could better scale out in a cloud environment.

5.3.3. Changing the Delivery Process

Comptel InstantLink is tested using continuous integration and additional performance and system testing, but they are not yet part of the automated pipeline. Continuous delivery could be reached by extending the automated test environments to test the whole deliverable solution using real customer logics and configurations. Customer environments are highly customized and complex, which makes it difficult to find a representative case. A version control system for the configuration files should be implemented, since this would also benefit delivery projects.

It should be easy to find which versions of each product were deployed together, and what were the configurations for each of them. When everything would be easily available, a new version could be configured by making necessary changes to the existing settings, and deployed using, for example, blue green deployment. A further improvement, that would make cloud deployments easier, is to implement a Comptel Instantlink specific virtual machine, which functions as a basis of every installation.

6. SUMMARY

The load on OSS/BSS systems will grow in the future, as the amount of data and the number of Internet and mobile device users keeps growing. Networks will also be more dynamic due to new technologies, such as NFV and SDN. OSS/BSS systems would benefit from the scalable resources provided by cloud environments. However, migrating to a cloud environment is not an easy task. The possible problems are not always transparent, because cloud environments are relatively new and marketing focuses on the benefits.

The goal of this thesis was to create an easy to use method for assessing the feasibility of deploying a software product to a cloud environment, and to analyze the cloud compatibility of Comptel InstantLink. The benefits and problems of cloud environments were gathered from literature, and used as the basis for the presented method.

The goals of the thesis were met. The method can be used to evaluate the main questions, when a company wants to deploy their application to cloud. The first diagram helps in analyzing, whether there is a need for a cloud environment. The second continues to evaluate how well the application suits cloud environments. The third diagram helps in evaluating the main problems in the existing software architecture. There is still room for improvement, and especially the third diagram would need to be changed. Implementing a web service based on the method, would make it easier to use and it could provide an output that includes additional information.

The analysis of Comptel InstantLink showed that deploying to a cloud environment is not yet a feasible option. However, the current direction is promising and Comptel InstantLink is heading towards a more cloud compatible architecture. Main concerns are the usage of the file system for logging, a single database, and a single component handling request locking, which hinders scalability.

REFERENCES

[1] R. Buyya, S. Chee, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25:599–616, 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, Stoica I., and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical report, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, February 2009.

[3] P. Mell and T. Grance. The NIST definition of Cloud Computing. *Recommendations of the National Institute of Standards and Technology*, September 2011.

[4] Amazon Web Services. Amazon EC2 - Virtual Server Hosting. Available (22.01.2016): `http://aws.amazon.com/ec2/`.

[5] Google. Google App Engine. Available (22.01.2016): `https://cloud.google.com/appengine/`.

[6] Microsoft. Microsoft Azure Documentation Center. Available (27.01.2016): `https://azure.microsoft.com/en-us/documentation/`.

[7] Google. Google Compute Engine. Available (22.01.2016): `https://cloud.google.com/compute/`.

[8] Amazon Web Services. AWS Elastic Beanstalk. Available (22.01.2016): `https://aws.amazon.com/elasticbeanstalk/`.

[9] Rackspace. Rackspace Private Cloud Software. Available (20.05.2016): `https://www.rackspace.com/cloud/private/openstack/script`.

[10] B. Deeter and R. Jung. Software as a Service Pricing Strategies. Technical report, Bessemer Venture Partners, July 2013.

[11] J. Villanueva. Active-Active vs Active-Passive High Availability Cluster, 2015. Available (18.05.2016): http://www.jscape.com/blog/active-active-vs-active-passive-high-availability-cluster.

[12] S. Barker and P. Shenoy. Empirical evaluation of latency-sensitive application performance in the cloud. In *MMSys '10 Proceedings of the first annual ACM SIGMM*, pages 35–46. ACM, 2010.

[13] W. Jansen and T. Grance. Guidelines on Security and Privacy in Public Cloud Computing. *Recommendations of the National Institute of Standards and Technology*, December 2011.

[14] Y. Kanemasa, Q. Wang, J. Li, M. Matsubara, and C. Pu. Revisiting Performance Interference among Consolidated n-Tier Applications: Sharing is Better Than Isolation. In *International Conference on Services Computing*, pages 136–143. IEEE, 2013.

[15] C. Lai, Q. Wang, J. Kimball, J. Li, J. Park, and C. Pu. IO Performance Interference among Consolidated n-Tier Applications: Sharing is Better than Isolation for Disks. In *International Conference on Cloud Computing*, pages 24–31. IEEE, 2014.

[16] V. Persico, P. Marchetta, A. Botta, and A. Pescapé. Measuring Network Throughput in the Cloud: the case of Amazon EC2. *Computer Networks, Special Issue on Cloud Networking and Communications II*, 93:408–422, 2015.

[17] V. Persico, P. Marchetta, A. Botta, and A. Pescapé. On Network Throughput Variability in Microsoft Azure Cloud. In *IEEE GLOBECOM 2015*. IEEE, 2010.

[18] Finlex. Henkilötietolaki, 1999. Available (23.02.2016): http://www.finlex.fi/fi/laki/ajantasa/1999/19990523.

[19] Finnish Communications Regulatory Authority. Pilvipalveluiden turvallisuus, 2014. Available (22.02.2016): https://www.viestintavirasto.fi/ohjausjavalvonta/ohjeetjajulkaisut/ohjeidentulkintojensuositustenjaselvitystenasiakirjat/pilvipalvelujenturvallisu

us-mitaorganisaatioidentulisihuomioidapilvipalvelujahyodyntaessaan.html.

[20] Data Protection Ombudsman. Henkilötietojen ulkomaille luovutus, 2015. Available (23.02.2016): http://www.tietosuoja.fi/fi/index/rekisterinpitajalle/ilmoitusvelvollisuus/henkilotietojenulkomailleluovutus.html.

[21] European Comission. Restoring trust in transatlantic data flows through strong safeguards: European Commission presents EU-U.S. Privacy Shield, 2016. Available (02.03.2016): http://europa.eu/rapid/press-release_IP-16-433_en.htm.

[22] J. van Hoboken, A. Arnbak, and N. van Eijk. Cloud Computing in Higher Education and Research Institutions and the USA Patriot Act. *Social Science Electronic Publishing*, 2012.

[23] Oracle. Licensing Oracle Software in the Cloud Computing Environment, 2013. Available (24.03.2016): LicensingOracleSoftwareintheCloudComputingEnvironment.

[24] Oracle. Oracle Partitioning Policy, 2015. Available (24.03.2016): http://www.oracle.com/us/corporate/pricing/partitioning-070609.pdf.

[25] G. Semones. Building cloud-ready, multicore-friendly applications, 2009. Available (04.12.2015): http://www.javaworld.com/article/2078001/java-concurrency/building-cloud-ready--multicore-friendly-applications--part-1--design-principles.html.

[26] J. Humble, C. Read, and D. North. The Deployment Production Line. In *Proceedings of the conference on AGILE 2006*, pages 113–118. IEEE, 2006.

[27] J. Humble and D. Farley. *Continuous Delivery - Reliable Software Releases through Build, Test, and Deployment Automation*. Pearson Education, 2010.

[28] D. Sato. CanaryRelease, 2014. Available (19.05.2016): http://martinfowler.com/bliki/CanaryRelease.html.

[29] Amazon Web Services. Scenario 3: VPC with Public and Private Subnets and Hardware VPN Access. Available (19.05.2016): http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Scenario3.html.

[30] B. Doerrfeld. How To Control User Identity Within Microservices, 2015. Available (19.05.2016): http://nordicapis.com/how-to-control-user-identity-within-microservices/.

[31] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of Carlifornia, 2000.

[32] H. He. What Is Service-Oriented Architecture, 2003. Available (30.10.2015): http://www.xml.com/pub/a/ws/2003/09/30/soa.html.

[33] J. Lewis and M. Fowler. Microservices, a definition of this new architectural term, 2014. Available (30.10.2015): http://martinfowler.com/articles/microservices.html.

[34] S. Peyrott. An Introduction to Microservices, Part 3: The Service Registry, 2015. Available (11.05.2016): https://auth0.com/blog/2015/10/02/an-introduction-to-microservices-part-3-the-service-registry/.

[35] B. Adler. Architecting Scalable Applications in the Cloud: The Caching Tier, 2012. Available (02.05.2016): http://www.rightscale.com/blog/enterprise-cloud-strategies/architecting-scalable-applications-cloud-caching-tier.

[36] D. Chiu and G. Agrawal. Evaluating caching and storage options on the Amazon Web Services Cloud. In *2010 11th IEEE/ACM International Conference on Grid Computing*, pages 17–24. IEEE, 2010.

[37] Comptel Corporation. Comptel InstantLink Functional Description, 2015. Confidential.

[38] R. Lahtinen, 2016. private communication.

[39] F. Ruhl, S. Steiff, S. Latava, P. Moore, J. Hawkins, T. Cauble, and G. Feeney. Impact of SDN and NFV on OSS/BSS. Technical report, Open Networking Foundation, March 2016.

[40] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*, 16, 2014.

[41] radisys. 3g/HSPA product. Available (28.9.2015): `http://www.radisys.com/products/trillium/3g-hspa/`.

[42] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, J. Benitez, U. Michel, H. Damker, K. Ogaki, T. Matsuzaki, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Kolias, I. Guadini, E. Demaria, R. Minerva, A. Manzalini, D. López, F. Salguero, F. Ruhl, and P. Sen. Network Functions Virtualization - An Introduction, Benefits, Enablers, Challenges & Call for Action. Technical report, October 2012.

[43] M. Dence, 2016. private communication.