SAKU SUHONEN

CLOUDIFICATION OF REAL TIME NETWORK MONITORING PRODUCT

Master of Science thesis

Examiner: Prof. Timo D. Hämäläinen.
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 5th November 2014

TIIVISTELMÄ

SAKU SUHONEN: Reaaliaikaisen verkon monitorointiohjelman pilveytys
Tampereen teknillinen yliopisto
Diplomityö, 52 sivua
Toukokuu 2015
Sähkötekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Ohjelmoitavat alustat ja laitteet
Tarkastaja: professori Timo D. Hämäläinen, DI Timo Vesterinen

Avainsanat: pilvi, cloud, cloudification, OpenStack, ETSI

Nykyään lähes kaikki ohjelmistot halutaan siirtää pilveen. Syitä ohjelmien siirtämiseen pilveen on monia, kuten kulujen pienentäminen ja skaalautuvuus. Työssä käsiteltävä reaaliaikainen verkon monitorointiohjelma on tähän asti asennettu fyysisille servereille, mutta asiakkaat haluavat ohjelmistojensa toimivan heidän omien konesaliensa pilvissä. Tässä työssä tämä monitorointiohjelma ja siihen kuuluvia komponentteja pilveytettiin.

Tämän työn alussa käsitellään ensin teoriaa pilvestä ja virtualisoinnista. Seuraavaksi käsitellään kommunikaatioverkkojen toimintojen virtualisointia ajavan ETSI NFV ISG:in ajamaa VNF konseptia. Tämän jälkeen käydään läpi pilveytettävän verkkojen monitorointiohjelman perustoiminnat ja ominaisuuksien evaluointi. Lopuksi käsitellään pilveytyksessä käytetyt ohjelmistot ja itse pilveytys. Pilveytykseen kuului monitorointiohjelman asennus ja toiminnallisuuden varmistus OpenStack-pilvialustalla. Pilveen asennettavan virtuaalikoneen asennuskuvan teko automatisoitiin tulevan kehitystyön helpottamiseksi. Mahdollisia jatkotoimenpiteitä kehitettiin ja analysoitiin, jotta tulevaisuudessa saataisiin mahdollisimman suuri hyöty pilven käytöstä.

Työn tavoitteeseen päästiin, koska monitorointiohjelma toimii nyt pilviympäristössä. Pilviympäristön kaikkia hyötyjä ei päästä käyttämään, koska ohjelmiston arkkitehtuuri ei nykyisessä muodossaan toteuta pilviohjelmiston tyypillisiä ominaisuuksia. Ohjelmiston kypsyys ja koko, joka on noin 500 000 riviä koodia, ei nopealla aikataululla pystytä muuttamaan niin että pilviympäristön kaikki hyödyt pystyttäisiin ottamaan käyttöön.

ABSTRACT

SAKU SUHONEN: Cloudification of real time network monitoring product
Tampere University of Technology
Master of Science Thesis, 52 pages
May 2015
Master's Degree Programme in Electrical Engineering
Major: Programmable Platforms and Devices
Examiner: Professor Timo D. Hämäläinen, MSc Timo Vesterinen

Keywords: pilvi, cloud, cloudification, OpenStack, ETSI

Currently, almost all applications are wanted to move to the cloud. There are many reasons to move applications to the cloud, for example decreasing costs and scaling capabilities of the cloud. In this thesis the real time network monitoring product is until now installed on physical servers, but customers are requesting the possibility to install it into their own cloud in their datacentres. This is the reason why this network monitoring product and components belonging to it are now cloudified.

In the beginning of this thesis the theory of the cloud and virtualisation is explained. Then the virtualisation of network functions driven by ETSI NFV ISG is explained. After this the basics of the network monitoring product is described with the evaluation of its characteristics. Finally the programs used for the cloudification are described and the cloudification itselt is explained. Cloudification included installation of the monitoring product and verification of the functionalities on the OpenStack cloud operating system. Cloud image creation was automated during the thesis to help in the future development. Possible improvements to benefit more from using a cloud environment were developed and analysed.

The target in this thesis was achieved because the monitoring product is now functional in cloud environment. All of the benefits of the cloud are not achieved because the architecture of the product does not fulfil all the properties of a cloud product. The maturity and the size of the product make it complex to achieve all of the benefits of the cloud in a fast schedule. The product size is about 500 000 lines of code.

PREFACE

Great thanks to my mentor Timo Vesterinen for guiding on the thesis work and other colleagues for the good support. Thanks for the management for this opportunity to finalize my studies. Thanks for Timo Hämäläinen for guidance and help during the thesis project.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application Programming Interface
BSS	Business Support System
CAPEX	Capital Expenditures
CLI	Command-line interpreter/interface
COTS	Commercial Of The Shelf
CPU	Central Processing Unit
CSP	Communications service provider
DAS	Directly attached storage
EMS	Element Management System
ETSI	European Telecommunications Standards Institute
GUI	Graphical user interface
HA	High Availability
IaaS	Infrastructure as a Service
ISG	Industry Specification Group
IT	Information technology
KVM	Kernel-based Virtual Machine
MBR	Master boot record
NE	Network Element
NFVI	Network Functions Virtualisation Infrastructure
NFVI-PoP	NFVI Point of Presence
NFV	Network Functions Virtualisation
NIST	National Institute of Standards and Technology
NUMA	Non-uniform memory access
OPEX	Operational Expenditures
OSS	Operations Support System
OS	Operating System
PaaS	Platform as a Service
RHEL	Red Hat Enterprise Linux
SaaS	Software as a Service
SDN	Software defined networking
SLA	Service-level agreement
Sysprep	System Preparation
TTY	Tampereen teknillinen yliopisto
TUT	Tampere University of Technology
VIM	Virtualised Infrastructure Manager
VM	Virtual Machine
VNF	Virtualised Network Function

1. INTRODUCTION

This thesis is written to discover possibilities to enhance a mature, well established, and real-time network monitoring product, later referred to as “the monitoring product”, to support information technology (IT) cloud computing infrastructure. This particular product will be virtualised by the time the thesis is ready, so the next evolution step for the product is to move to a cloud.

Figure 1 visualizes an example of a network infrastructure monitored by the product. When a mobile device is communicating with the network, it creates data traffic in the network. This traffic consists of user data and element management data. The monitoring product monitors the network using the data coming from the network elements.

Benefits for the vendor are selling the monitoring product as software-only without dedicated hardware bundled with the product. This creates capital expenditure (CAPEX) savings and fastens the deployment time. Use of cloud environments is also requested by the customers. The vendor has to stay up with the demands of the market and technology advancements as everything seems to be going to the cloud, including the network elements (NE) monitored by the product.

The aim of this thesis is to study what are the requirements and what needs to be done to get the monitoring product to perform its tasks in the cloud environment. Possible future improvements to get most of the benefits of the cloud are also evaluated. This creates lots of cumbersome problems to be solved for a mature product, because in cloud environment products need to be more dynamic and highly automated to get most of the cloud. Reasons to go in to the cloud are cost savings for customers. Also, the monitored network elements are going in to the cloud.

This work is based on the standard Network Functions Virtualisation (NFV) specified by Industry Specification Group (ISG) under the European Telecommunications Standards Institute (ETSI). Their research results and publications are publicly available in [14]. Target is to follow ETSI NFV implementation, because the vendor, customers and competitors are members of the ETSI NFV. At the time of writing this thesis not all of the work or documentations of ETSI NFV have been finalized, but some of the documents are available as a draft version.

Chapter 2 introduces the basics of a cloud. Virtualisation will be covered in the subsequent sections as it is one of the basis of the benefits in the cloud. Some familiar cloud services are described and explained as those are most visible for most of the people and the usage of cloud services are growing. Also, the benefits of the cloud compared to a dedicated hardware are described.

Cloudification will be covered in the Chapter three. The best practices that support cloudification and creating new cloud based products are gathered from [11][16][23][39]. OpenStack's best practices [11] were chosen, because OpenStack is used in this thesis. Kris Jamsa [23] evaluates aspects of migrating to the cloud on the point of view of existing application with real life cases. Amazon's best practices [39] were chosen, because Amazon is a big cloud provider. Additional aspect was found from Fehling et al [16]. Best practices are accompanied with pitfalls of cloudification when moving product to the cloud.

Chapter four will consist of the way how ETSI NFV has proposed to implement the network functionalities to virtual and cloud infrastructure. First, there will be covered the organization itself and its objectives. Also, the way the clouds and virtualised environments should be managed to achieve good multi-vendor interconnectable cloud appliances.

The product under the study is covered in Chapter five. The product will be described at high level to understand the basics of the monitoring product. Feasibility evaluation for the cloud is in this chapter. How the best practices of cloud could be implemented to the product are also evaluated.

Cloudification itself is presented in chapter six. This also includes the used cloud environment and all the products and applications that were required for a complete prototype. Conclusions are given in the last Chapter.

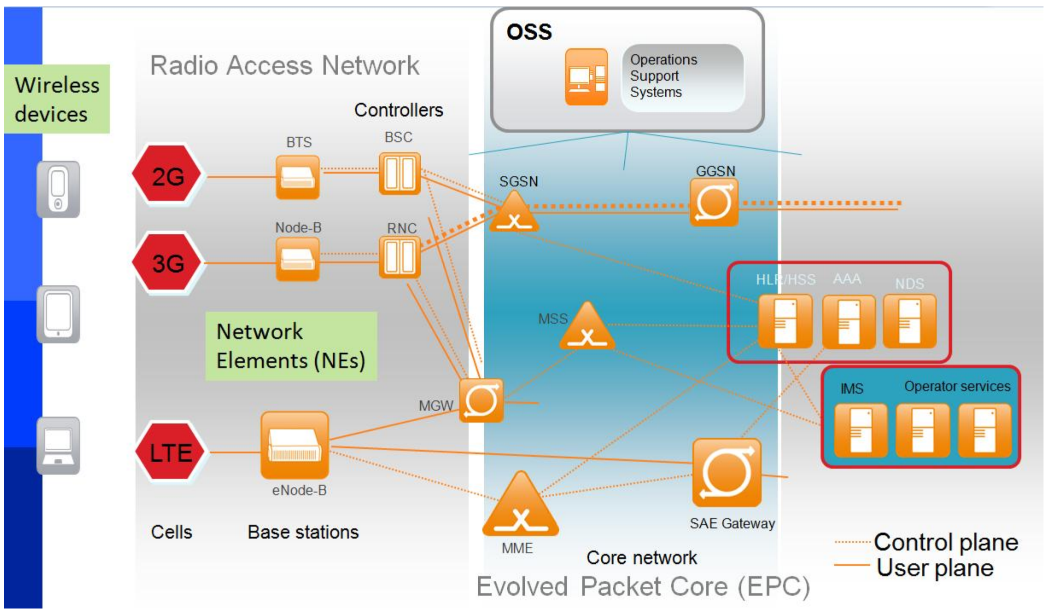

Figure 1: Cellular network infrastructure.

2. BASICS OF A CLOUD

Cloud is the next step after virtualization, where software (SW) is created independent of the underlying hardware (HW). In the next subchapters there is introduction to virtualization and basics of cloud. One of the main ideas of cloud is to be able to use commercial off the shelf (COTS) hardware like in virtualization, but cloud adds the automated management and elasticity in to the environment. Using COTS hardware will drop the CAPEX of the infrastructure provider as there would not be needed for a variety of different HW. Use of COTS will also decrease operational expenditure (OPEX) in a sense that then there would not be required so many different HW specialist for maintenance. The user changing from own private datacentre to cloud infrastructure provided by some other company would decrease CAPEX to more OPEX oriented [16]. Even though virtualisation is not mandatory for cloud, it is considered to be the underlying technology in this thesis [30].

National Institute of Standards and Technology (NIST), which is an agency of the U.S. Department of Commerce, describes cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. The cloud description follows definitions from NIST in this work. [27]

There are four main types of clouds: public, private, community and hybrid. Public cloud provider offers virtual machines (VM) and computing resources for anyone. Private persons and companies can buy resources and services from the cloud provider and add their own input or product on top of it. This can be then sold forward or used by the buyer itself.

Private cloud can be inside company premises, but used only by the company for their internal functions. Private cloud can also be inside cloud provider premises, but completely separated from other clouds so that specific physical servers are reserved for the offered private cloud. This kind of private cloud offering is a good option for small companies that do not have resources to build their own, but require one because of sensitive data stored or processed in the cloud. In the community cloud, the infrastructure is provisioned for specific community of companies that have similar requirements and security concerns for the cloud. Community cloud can be owned and operated by one or more of the companies in the community or it can be outsourced to a third party. Hybrid cloud is a mixture of two or more of other cloud types. [27]

The five most important cloud definitions are:

- On-demand capabilities: Users can acquire more computing resources from cloud easily without special expertise or help from the cloud administrator.
- Broad network access: Access to the resources in the cloud is available over the network using any device.
- Resource pooling: Physical computing resources are shared with two or more user groups also known as multi-tenancy.
- Rapid elasticity: Resources are allocated and released on-demand or automatically. For high usage peaks the cloud allocates more resources for the application or service.
- Measured service: All the usage is charged by hourly basis or the billing is based on the actual resource usage.

These characteristics [36][27] define more cloud service offerings for cloud users rather than management of the cloud owner. As the product studied in this documentation is going to be deployed in a private cloud, and the cloud provider and the users of the monitoring product are in the same company, few more cloud features are considered:

- Lower costs: Physical server utilization is higher. Two or more applications can run on same physical server [36]. Using of COTS compared to specific HW.
- Reliability: The ability of cloud computing to provide load balancing and failover makes it highly reliable [36]. Multiple redundant sites creates efficient disaster recovery for computing.

Some other common aspects and motivations for companies to move to cloud are: cost of creating a new server facility compared to renting the infrastructure from a cloud provider, and possibility to rent more computing power in occasional situations. This kind of situation could be, for example, the launch of a new webpage with predicted high load in the first hours or for a company website that is publishing a new product.

2.1 Virtual environment

In virtualisation the underlying hardware components are emulated as smaller virtual components to virtual machines. This is done by virtual machine monitor, which is also called hypervisor. These virtual machines are independent in relation to each other so that the program running on one virtual machine will not affect the program running on a another virtual machine. In comparison of having one physical server with one application, virtualisation allows multiple virtual servers on one physical server. Each virtual server can have their own operating system (OS) and applications. Operating system on virtual machine is called guest OS. [4]

Virtualisation can be divided in two levels: Full virtualization and Paravirtualisation. In the full virtualization, the whole underlying hardware has to be emulated. In paravirtualisation, the guest OS is aware of the hypervisor and not all of the hardware need to be emulated. To make guest OS aware of the hypervisor, the kernel of the guest OS needs to be modified. This is currently possible only on open-source OSes and thus not possible for Windows® operating systems. Paravirtualisation can attain better performance than full virtualization, because it does not need full emulation of the hardware. [4]

Virtualisation and virtual machines can be done with hypervisor, which is implemented directly on top of physical hardware or it can be implemented on top of OS. Hypervisor on top of hardware is called Bare-metal or Type-1 hypervisor. Hypervisor on top of OS is called Hosted or Type-2 hypervisor. Both cases are illustrated in Figure 2 below.

Creating virtual environment on top of OS diminish the profits of virtualisation as the OS uses resources and those resources could be used by VMs. Exception for this is Operating-system-level virtualization, where the kernel of an operating system allows isolated user space instances also known as containers. Example hypervisor products that support full virtualization [4] are Linux's Kernel-based Virtual Machine (KVM), VMware ESX/ESXi (Elastic Sky X*)* and Microsoft Hyper-V. Xen is one example of a hypervisor that supports paravirtualisation [4].

The sum of virtual resources from one physical server can exceed the sum of physical resources [9]. This overcommitting or over-provisioning of resources can be useful in situations where the programs on VMs are not using all the time all of the reserved resources. If the resource consumption starts to rise, some of the VMs can be moved to another physical server manually or with cloud manager application. The possibility to overcommit depends on the used hypervisor on the host machine [34].

In virtualised environment the physical machine is called *host* and the virtual machine is called *guest*. Similarly if the host machine has OS under the virtualisation layer that OS is called host OS as seen in Figure 2.

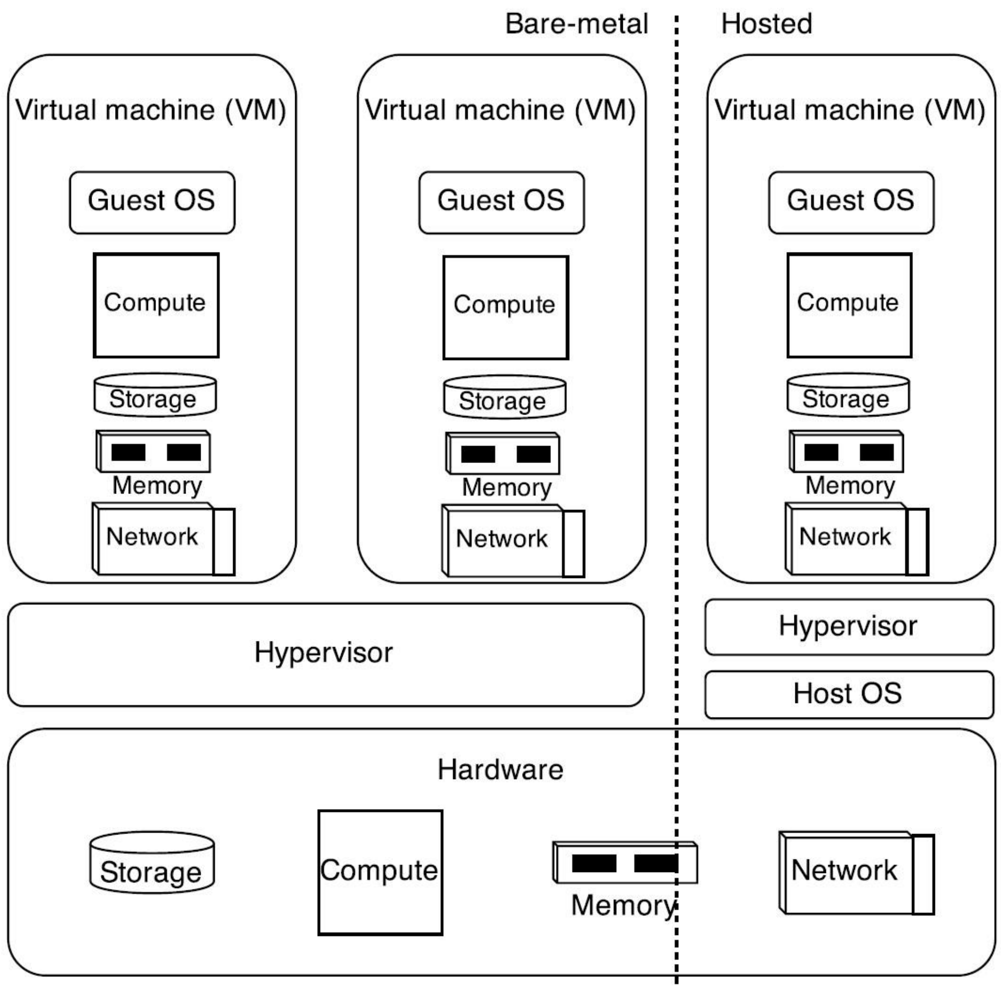

Figure 2: Virtualisation layers on bare-metal and on hosted.

Overhead caused by virtualisation depends on overall activity of the application, other VMs activities and how the environment is configured. According to Grandinetti et al. the performance slowdown caused by virtualisation is in general about 2%. [19]

The current hypervisor functionalities and features are increasingly implemented in hardware and firmware, which means that hypervisors are not the same in future or they might have disappeared. [19]

2.2 Cloud service models

Cloud systems are divided in three main systems as the software, platform and infra-structure (SPI) model. These systems are referred to as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) as seen in Figure 3. There are more similar as-a-service abbreviations, but aforementioned three are the most commonly used. All of these as-a-service models can be collectively referred to as anything-as-a-service (XaaS).

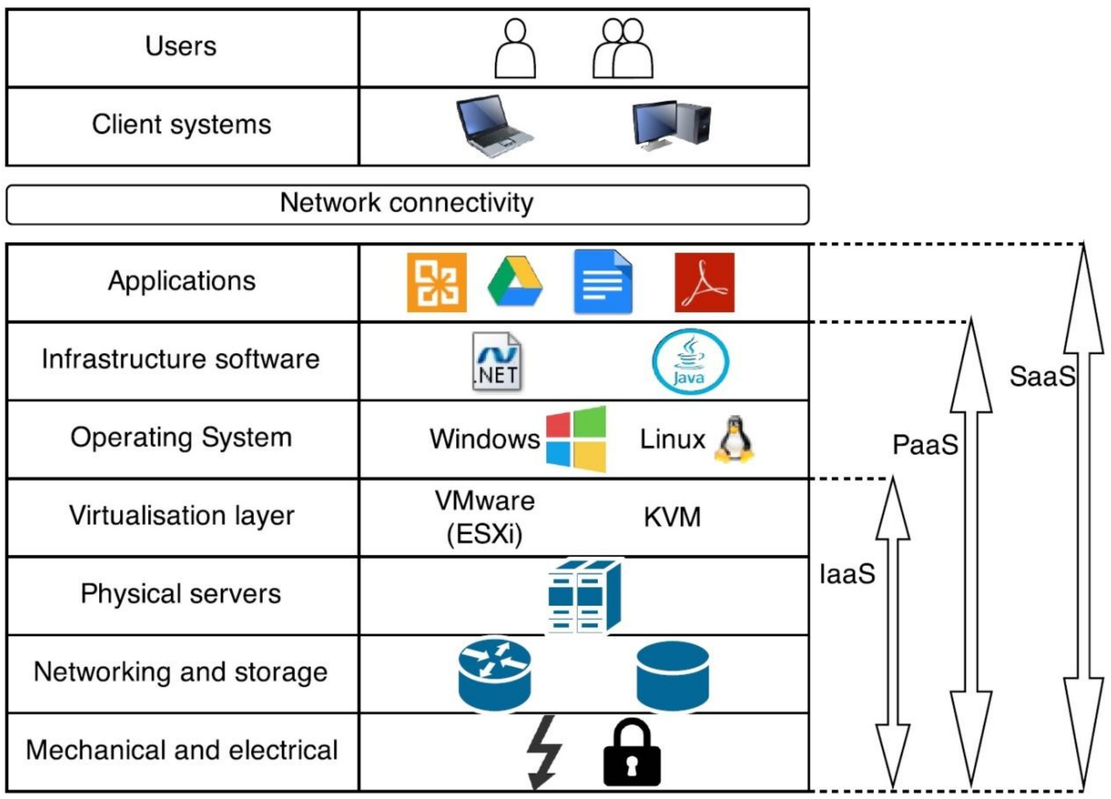

Figure 3: SPI service models adapted from [30].

SaaS is the most used cloud service as it is targeted to end users. One example product of SaaS is Google Docs, but users might not realize that they are using a cloud service. SaaS applications are mostly accessed through browsers, which transforms the required computing power from the users' computer to the clouds' infrastructure, from where the application is provided. Upgrades and fixes to the applications do not require user actions as these are done at the cloud by the provider without disrupting users. These characteristics make SaaS applications attractive alternatives to desktop applications. Usually these services have usage based costs. [1]

In PaaS, the provider supplies the user with a platform of software environments and application programming interfaces (APIs) to interact with the environment. PaaS is mostly used by developers to develop applications to the cloud. The PaaS model can provide additional services to make application development easier. For example, authentication services and user interface components can be provided by the PaaS environment. Google's App Engine is one example of PaaS, which provides Python and Java runtime environments. Google's App Engine also provides different services, like authentication, to be integrated to the application to make the development of the application faster. [1] Amazon offers different types of software to be used and licensed on an hourly or monthly basis cost on the VMs bought from their cloud service [3].

IaaS model provides virtual computer instances are called virtual machines. IaaS provider can create multiple VMs on top of one physical server. This model allows IaaS provider to utilize more efficiently underutilized physical hardware in a datacentre by exploitation of both time division and statistical multiplexing. Amazon's Elastic Compute Cloud (EC2) is one example of IaaS, where users can have different sized VMs. [1] The IaaS model is most relevant to this thesis as the monitoring product is deployed similarly. Communications Service Provider (CSP) would provide the infrastructure and virtual server with OS, which is Microsoft Windows® in this case.

In this environment, the monitoring product would be delivered only as software without HW or OS, in contrast to the more traditional bare-metal deployment where everything is delivered. Then the operations personnel would use this product from their cloud. One future target is to provide and sell only software, as this will give the CSP the flexibility to choose third party hardware provider and means to acquire the OS.

Figure 3 shows the differences between IaaS, Paas and SaaS providers, even though the user might not have visibility to all of the layers. SaaS provider offers everything from application to physical parts of the cloud. Figure 4 below describes examples of responsibilities of different XaaS providers according to Figure 3 above.

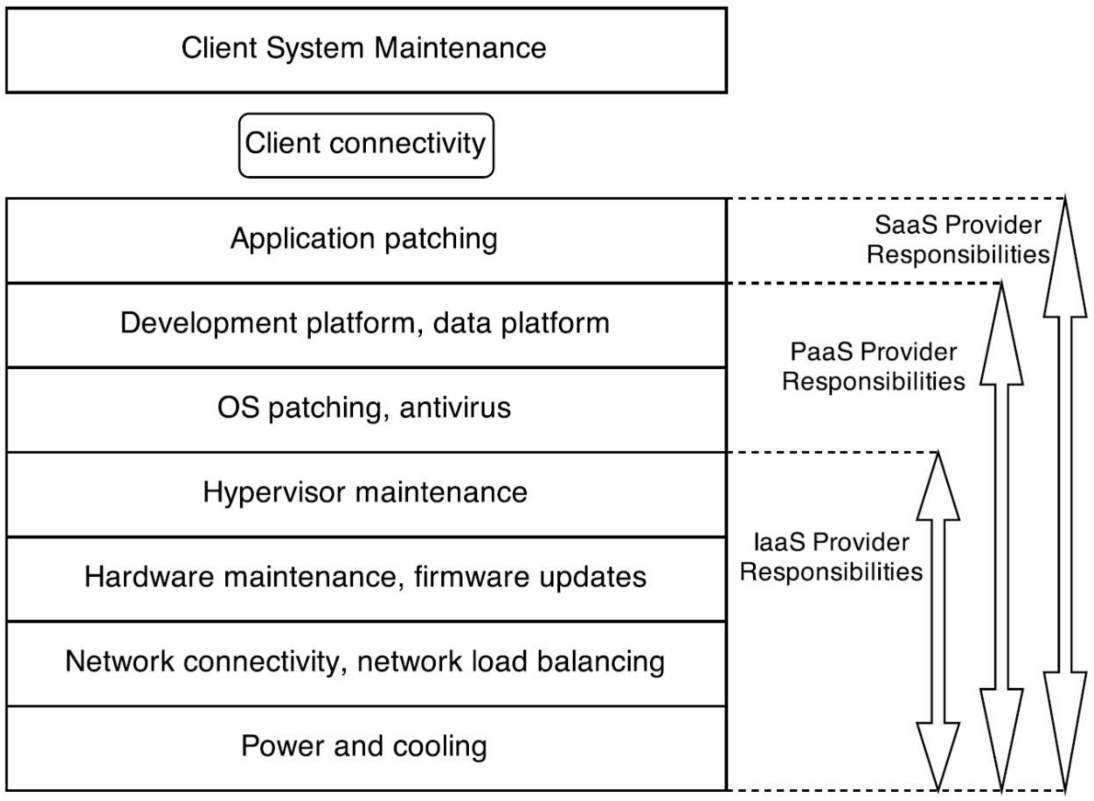
Client System Maintenance
Client connectivity
Application patching
Development platform, data platform
OS patching, antivirus
Hypervisor maintenance
Hardware maintenance, firmware updates
Network connectivity, network load balancing
Power and cooling
SaaS Provider Responsibilities
PaaS Provider Responsibilities
IaaS Provider Responsibilities

Figure 4: SPI providers responsibilities adapted from [30].

As seen from Figure 4 by the user perspective, the service provider responsibilities depends on the type of service they are providing. But higher level service provider, for example SaaS provider might be buying a lower level service from other company and then adding their software/service on top of it to increase the value. In this case, a SaaS provider is not controlling the lower levels and service-level agreement (SLA) between SaaS and the lower level provider affects to the contracts or SLAs that SaaS offers to its customers.

2.3 Benefits of the cloud

Cloud computing brings quite a lot of benefits for individual users and companies, large and small. Individuals and companies can use cloud services for example to store or backup data like pictures.

a) Development cycle and time to market can be decreased. As mentioned earlier that PaaS providers might offer different pretested services and APIs to be implemented on the application to fasten development and this leads to faster time to market, when developing in cloud for cloud platform. [1] Developing application to cloud and using existing cloud environment, internal or external, can decrease time to market as there is no need to order and set up new server hardware to host the application.
b) Virtualisation is a base of most of the benefits that cloud brings. As stated previously virtualization allows running many applications on one physical server. This improves the utilization of hardware and decreases costs. Virtualisation also enables life cycle continuation of legacy applications. VM with specialized OS provides an ideal environment for legacy applications that are sensitive to the execution environment, and have working and validated codes without need to modify. Virtualisation also enables customized and reproducible environment to target a specific application, which can be used immediately or reused later. [1]
c) Almost zero upfront infrastructure investments is one of the most attractive benefit on moving to cloud services from the business point of view. Building up own large scale system is big investment in real estate, physical security, hardware, hardware management and personnel to operate the system. Hardware costs consist of racks, servers, routers and backup power supplies. Hardware management consist of power management and cooling. Old personnel would need to be also trained for the new environment or new personnel would need to be hired. [39]

Other business reasons to move to or start using cloud services are benefits of elasticity and scalability in the cloud. Traditionally scaling is done by investing on more powerful server or investing on more servers if the architecture of the application allows it. This scaling would be done by predicting the demand and investing early enough to get the hardware to keep up with the demand. Figure 5 illustrates traditional scaling approaches with predicted demand. Scale-up would be done by changing the server to a new improved and more powerful version causing huge investment at once. Scale-out would be done by adding more servers to distribute load causing smaller investments from time to time.

If the actual demand suddenly rises higher than the capacity, it would cause angry customers and maybe losing customers as seen in Figure 5. This would mean becoming a victim of one’s own success, because capacity could not keep up with the demand. But on the other hand, if the demand suddenly drops and a huge investment on hardware has been done, then there is much more underutilized capacity and bigger expenses to cause the company to go out of business, becoming a victim of one’s own failure. With cloud service, the scaling can be done automatically or manually with service providers API. Which is much faster than traditional ways of scaling and the company pays only the resources used and the cost follows the demand. [39]

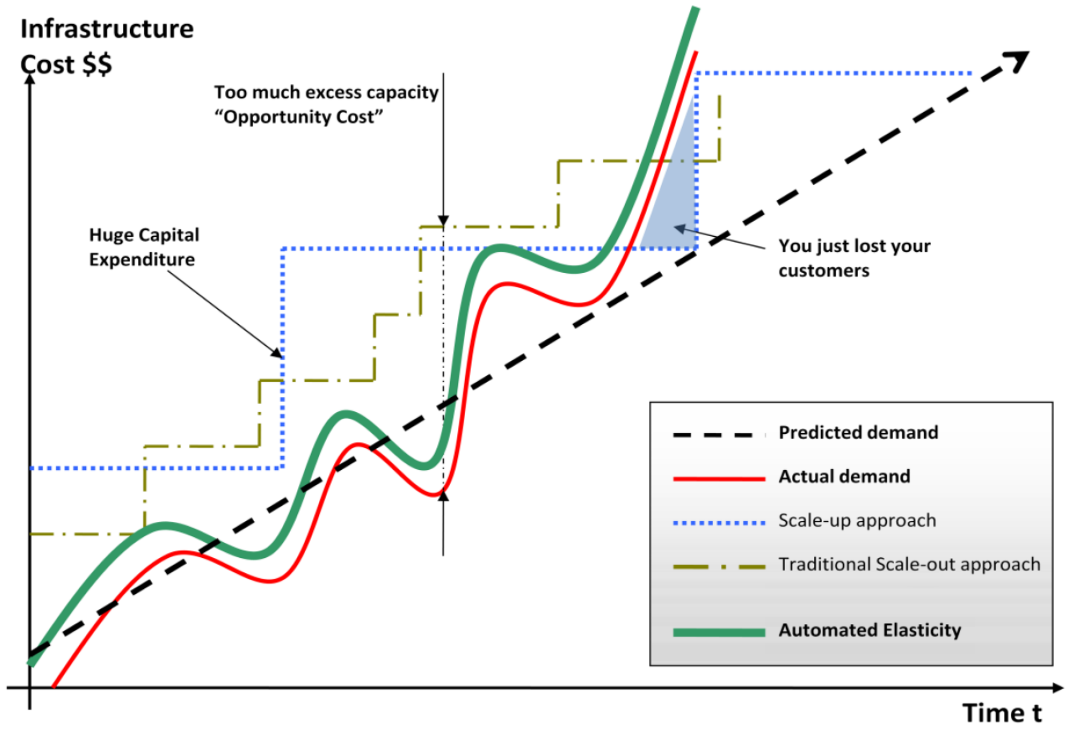

Figure 5: Automated Elasticity + Scalability [39].

Economies of scale in cloud computing makes the services provided cheaper. For example, a company could create email service internally for its employees with a cost of 25 cents per mailbox for 5000 employees. A cloud service provider can implement an email service for 100 000 users with 10 cents per mailbox and sell it forward as a service at 15 cents per mailbox. Therefore, outsourcing the email service benefits both participants. [30]

Reliability can cost a lot for an internal datacentre. Many cloud providers offers multisite location for a recovery plan if something happens in one of the sites. [30] With economies of scale, the backup within datacentre or locations is much cheaper for a cloud provider than for companies by themselves.

Cloud technologies and cloud service providers can leverage a market between large and small startup companies. Large companies had leverage to be able to make big investment on computational resources to start a new business, but nowadays cloud computing gives the possibility for small companies to enter the market without making huge CAPEX investments by paying for the cloud service provider by the used resources. [4] New companies might even have leverage as they do not have burden of the legacy applications that need to be supported and maybe integrated into the cloud.

2.4 Drawbacks and challenges of clouds

In public clouds the infrastructure is owned by the cloud provider and all the hardware, software and data is in the cloud providers' premises. This might cause some legal problems on who owns the data. This concerns also private clouds that are hosted by another company and community clouds that are created in collaboration. [30] From security and privacy point of view, trusting that the cloud provider is securing its cloud environment and has trustful employees, have to be considered. The contract with the cloud provider should dictate access rights and possibilities of cloud providers' employees to the data in the cloud. There might also be limitation by the law on storing and transporting the data over the geographical borders. [4]

Creating an own private cloud can be very expensive, with all the upfront CAPEX. The infrastructure would need to support current and future needs, and peak times. [30] Creating a hybrid cloud to manage with possible sudden increase of demand might turn to be very complex to implement. A hybrid cloud adds together benefits and disadvantages of different cloud types, depending on the implementation. [30]

Cloud performance is dependent on the performance of virtualisation, which is dependent on how the underlying hardware, network and VM have been configured. Performance of virtual processor is close to the physical one, but the performance of virtual network is behind the physical network. [1]

Lag of adopted standards in the clouds makes clouds incompatible. Creating application for one cloud, might take huge effort to get it work with other clouds. Integrating and reengineering legacy application to work in cloud or to work with cloud applications might cause significant costs. [1]

Cloud service providers charge by the usage of the resources. This creates need of thorough evaluation of required resources and price from different vendors. This evaluation is often hard and can result in unexpected costs. [1] For example if there is an application that requires a lot of computational power to process specific data, the computational cost and storage of the data on cloud might be cheap, but using network to transfer the data to the cloud might be really expensive or even slow. Amazon offers a service to transfer data with physical devices to in and out of their cloud [5].

Elasticity implementation requires deep understanding of the requirements of the application. Elasticity might be hard to implement, because it requires predicting when to add more VMs. Also, the VMs are not instantly running and working. Depending on the delivery it might also require installation and configuration of application before the new VM is helping on elasticity. When implementing location awareness to a multi-location cloud, the cloud should know from where the requests come in order to be able to setup a new instance to a site closer to the source of requests. Also, the VMs that are communicating with each other cannot be too far from away. [28]

Using cloud service or application is greatly affected by the latency of network. Cloud applications are not usually as customizable and feature rich as a version deployed on-premises, which might cause disappointment to the user. [36]

3. CLOUDIFICATION

Basically cloudification means the transformation of a service or an application from a local installation to an installation on a virtual instance on a group of servers to be accessible through network [8]. Depending on the current state of the architecture, implementation and usage of the product, it will define how cumbersome or simple the cloudification will be. But also the desired level of automation of the workflow and added or improved scalability will affect to the amount of work needed for the cloudification.

Clouds can have a manager entity that handles automation and management of the cloud environment and applications in the cloud. One example of this kind of manager is open source Cloudify, which works with many different cloud implementations. Cloudify is described as Cloud Application Orchestrator, which can handle infrastructure setup, installation of applications, upgrades and auto-scaling among many other features [18]. For this kind of management to be able to work it is required for the managed application to have an API or some other possibility to install, control and configure it without GUI and human interaction.

3.1 Definition of cloudification

In this chapter the characteristics of cloud listed in chapter 2 are opened and explained in more details and how each one can be applied to the cloud. What are the responsibilities of the cloud manager and which are implemented in the product that is cloudified.

For a product to be fully cloud compliant, it needs to have functionality or a component or a way to scale performance automatically. This is called **elasticity** [16]. The possibilities to scale performance are scaling up or scaling out, and these are illustrated in Figure 6. To scale by adding or removing computing resources for the VM or moving the application to run on different sized VM is called to scale up and down. Scaling up is also called vertical scaling. Another possible way to scale is to have the system divided in different VMs so that the scaling can be done by adding and removing VMs. The whole system can also be in one VM and adding another VM would increase the performance of the system, either by helping the original VM or by taking part of the load through load balancing. This kind of scaling by adding instances is called horizontal scaling or scaling out. [23] Evaluating of the system performance and finding bottleneck of the performance is one possible way of finding a functionality to implement scalability.

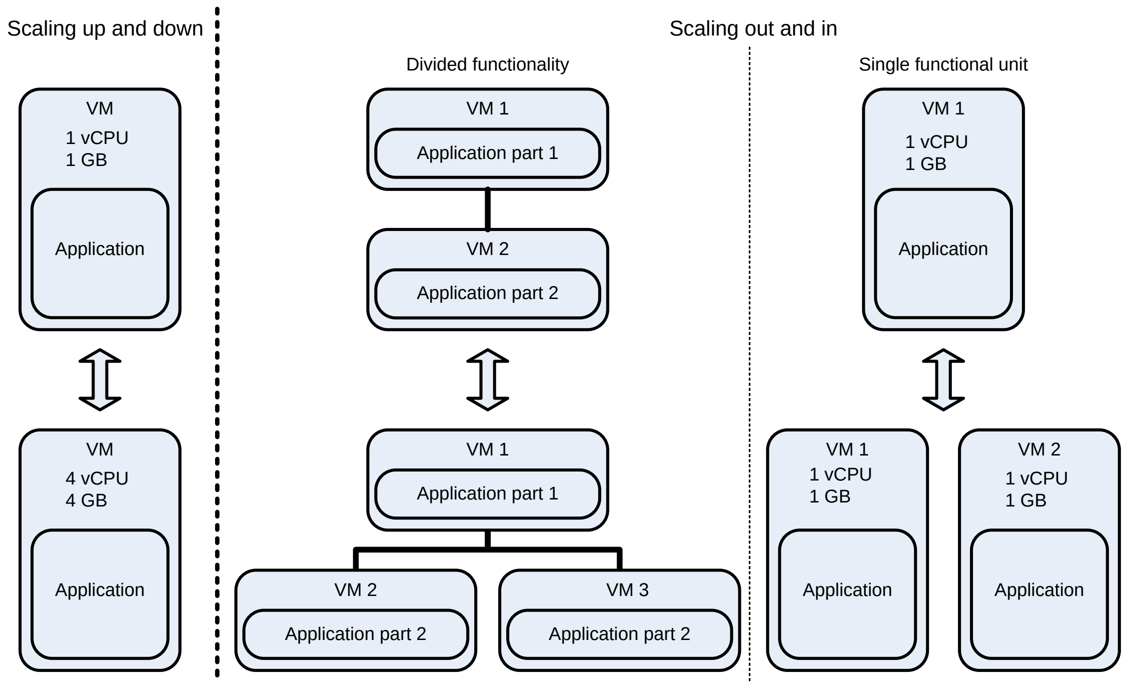

Figure 6: Possible ways of scaling an application in a cloud environment.

Scaling horizontally and using few servers costs less than using a single server that is few times better [23]. In a multiprocessor environment, if several processors attempt to access the same memory, the performance of the memory decreases. This can affect on a VM performance when the VM have more than one processor defined as VCPUs. This problem can be fixed with Non-uniform memory access (NUMA) technology that provides separate memory for each processor. Multiprocessor hardware is ideal for cloud infrastructure as each VM on the physical hardware can run on its own processor or core. [19] Processor affinity helps in infrastructure as one thread or VM would be bind to one or specific processors so that the change of computational processor would not affect the performance [31].

On-demand capability lets users to acquire more computing resources when they want. This can be done via command line or GUI by user or via API by the application itself [16]. Integrating local computational environment onto public cloud creates opportunity for on-demand scale-out for datacentres and small private clouds. This expansion to other cloud for resources is called cloudbursting. [6]

Broad network access is implemented so that cloud provider has acquired high-speed network access from the cloud to internet to support the traffic of all users [16]. Access is possible through any standard mechanism with any device, thin or thick client platform [27]. Common means to use cloud services are a common web browser and a proprietary application made for the service [36].

In **resource pooling** the cloud environment is shared between multiple customers. In public cloud these are different customers, but in private cloud these customers would be internal departments or projects that need separate allocation of resources in the cloud. Customer or internal group is often referred to as a tenant that has multiple users working for it. Each tenant has a share of the resources, which they can adjust according to their needs within the limits of the cloud infrastructure. [16]

Measured service. The use of resources for storage, processing and data exchange is measured to ensure transparency between the customer and the cloud provider in pay-per-use pricing model. So that the customer only pays for the service when they use it and only for the intensity in which they use it. This is enabler for many companies to shift CAPEX to OPEX in IT. This can be flexibly adjusted depending on the growth or decline of a business. [16]

Cost savings with a public cloud is clear with no high upfront investments, but a company internal private cloud can bring cost savings even though there is investments in setting up the environment. Centralizing and standardizing IT resources to enable automated management and easier maintenance of those resources can significantly reduce IT costs [16].

Regarding the reliability of cloud applications, it is possible to make them more reliable by load balancing, failover and availability. Load balancing helps to even load on different instances, but also it can provide functionality to provision and deprovision automatically without changing network configuration [4].

Availability means that the service is available to the users and the users can access to it. Business critical systems are usually referred as high availability (HA) systems, which means that planned and unplanned downtime of service is low as possible and the network connection to outside is functional with certain percentage value if it is affecting the use of the service. Availability is usually defined in SLA between customer and cloud provider. Availability might not include planned down times like maintenance, depending what is defined in the SLA. It depends on the SLA what is included in the availability and what is not, sometimes a planned downtime is not affecting to the agreed availability. SLA also defines fines or compensations depending on the level of failure in providing availability. When making an SLA it is good to have a clear definition and understanding on what are all the subjects agreed in SLA. It is also good to know who is responsible of the different components in the system if the cloud is acquired as service. [6]

A cloud can offer high availability, which means that the application is available almost all of the time to the user. This can be achieved with a failover solution where virtual servers are duplicated on another physical server in case the server breaks down [6]. Having more servers running for one application increases the IT resources used per application and decreases the costs savings, but for critical applications this is acceptable because the backup server, running or standby, takes immediately the place of the broken server and the users do not experience loss of data or usage of the application. Multi site back-up can be very important for some business as some disaster, natural or human made, can disable the whole datacentre containing the cloud. Availability is affected also by the network connection from cloud datacentre to internet [6].

3.2 Best practices for product on cloud

Additional thoughts and practices that should be considered after the definition of cloud and cloudification, mentioned in previous chapters, are collected from couple of sources in this chapter.

Kris Jamsa in his Cloud Computing [23] book describes aspects to consider when migrating existing application into the cloud, but also design aspects to keep in mind when creating a new application for the cloud. Some points summed and evaluated from his work related to this case:

Security related issues are one of the most important aspects to examine. Security concerns all the data stored in the cloud and transmitted data between user and cloud environment, especially if the cloud does not reside in the same premises as the users. Privacy concerns affect also the data, as who can access the data and how it is accessed. Also, what kind of data is on the servers and what kind of user rights are given for the users of the product. Access rights of cloud providers’ personnel have to be clearly defined to avoid misusage of the data. Concerning CSP, its user identification data have to be stored within the country of origin by the law in many countries. Many CSPs require that identification data usage have to be registered and the databases have to be encrypted to avoid possibility of misusage. Security affects everything and should be considered in every part of the system. Security concerns also patches and upgrades for the product and when and how they are implemented or installed. It is also good to find out what happens to the data and product in the cloud, if the cloud provider goes out of business. This comes to the definition on who owns the data in the cloud. [23]

Important design aspects are interoperability and portability for the product. Choosing cloud provider should not restrict the environment to just one cloud provider or to a one specific cloud. Used APIs from the provider might cause vendor lock-in unless the usage of APIs is designed well enough and are not bind to the critical functionality of the product. Depending on the target group, different client devices should be supported or at least the capability to add support for different devices should be implemented. Using open source software have to be evaluated, because open source does not exclude vendor lock-in. [23]

Preparing for disaster is almost mandatory in cloud environment. Clouds are meant to use COTS HW which is cheap and easily replaced. This leads to the fact that HW in cloud will break sooner or later. Recovery plans for different kind of disasters should be considered in the design phase. This can be done by examining disaster possibilities, and how they affect to business continuation, and how the effects of those disasters could be minimized. Clouds can provide good disaster recovery against natural disasters with multi-site cloud service. Recovery from HW failure can be done on local backups and a new instance will be created in case of failure. Cloud provider offers certain guarantees for the service they are providing and these are part of the SLA between companies. Stateless product can be easily recovered by launching a new instance, but a stateful product requires backup of the states. Designing a product or system so that it does not have a single point of failure creates better start for recovery, robustness, redundancy and reliability design. Disaster recovery planning is balancing between costs and risks. Risks evaluation should include threats from company personnel and cloud provider personnel; this can be part of security design as mentioned previously in this chapter. [23]

Maintenance is one of the most expensive phases of software life cycle. Cohesive and loosely coupled modules increases maintainability, code reuse, testability, and help in designing for redundancy. Maintenance design includes decisions about deployment; decision of OS, devices and browsers to be used for the product. Deployment of the product to the customer and delivery of upgrades and patches affects decisions on maintenance. [23]

Performance design includes evaluating potential bottlenecks and points to be optimized to gain better performance. Performance is affected by application characteristics like demand periods, average users, disk usage, database requirements and bandwidth consumption. Performance affects to the users perception of the product and a slow response of the product might drive away users. For a product that is cloudified is good to fully understand resource usage to be able to better estimate resource demand in the cloud. Clouds and cloud providers offer different applications to monitor the performance of the products in the cloud. [23]

Budget affects in every design aspect and decision because the budget limits the time and effort spend on improvements of the product, but inexpensive system deployment and development might be expensive to maintain. When moving to cloud, factors that affects budget are: the cost of current datacentre, payroll costs and costs of software licences. Datacentre costs consist of rent, power, cooling, colocation costs, server costs, data storage costs and network costs. Payroll costs for existing staff on current datacentre and how many would be required in case of cloud, whether it is own or outsourced. The costs of software licences might be lower in cloud-based environment. [23]

OpenStack have similar thoughts on their guidelines for designing an application for the cloud. Few key points from OpenStack designing guide [11]:

- Design for failure: Assume that everything fails and design backwards.
- Efficiency: Efficient designs become cheaper as they scale. Kill off unneeded components or capacity.
- Paranoid: Design for defense in depth and zero tolerance. Trust no one.
- Data management: Data is usually the most inflexible and complex area of a cloud.
- Automation: Leverage automation to increase consistency and quality and reduce response times.
- Divide and conquer: Make components as small and portable as possible. Use load balancing between layers.
- Elasticity: Increasing and decreasing should result in a proportional increase or decrease in performance and scalability.
- Dynamic: Enable dynamic configuration changes to adapt changing environment.
- Stay close: Move highly interactive components near each other to reduce latency.
- Loose coupling: Service interfaces, APIs and modularity to improve flexibility.

A good way for programming software for a cloud is the assumption that there are no special or specific hardware components. All hardware usage should be handled through common API/drivers and not doing any performance improvements relying on specific hardware [23]. This is important for the datacentre personnel because the cloud should use COTS hardware. Adding different hardware to the cloud infrastructure increases complexity of maintenance and operations because of required competence of datacentre personnel for that hardware.

Some clouds offer special hardware components or functionalities. For example Amazon offers different kind of instances for different purposes. They have general purpose instances, instance for graphic intensive calculation that has graphical processing unit and instance for high random I/O performance with SSD storage. [2]

Software architecture design is an important part of software system production. Using design patterns and architecture frames, designs and patterns. Identifying bottlenecks and scalable components in the system is important. The possibility to add new VM automatically to the system to increase computing power for high usage peak and removing afterwards is one way of exploiting elasticity.

The application in the cloud should be as stateless as possible, because session and application state might impact to the applications' ability to scale out. Stateless application or component can be added and removed more easily as state information does not need to be moved or copied from an old instance to a new one. [16]

Loosely coupled system improves scalability. One possibility to make system loosely coupled is to implement queues or buffers in the system. These queues or buffers are implemented between components to connect them together. If a component gets temporarily unavailable, the system buffers the data until the component is available again. [39]

3.3 Cloudification pitfalls

Reasons for failure on moving to the cloud can happen when a company fails to fully understand or embrace new technology. Rushing into development mode before architecture and design steps can lead to failure. Sometimes companies have unrealistic expectations like too aggressive due dates, too large of a scope and not the right people to perform the task. [24]

A common mistake is to think that migrating current application to cloud is just driving down the costs, but just few applications are good candidates to move to the cloud with the current architecture. It is common that the application is highly dependent on the physical HW and even on the technology stack that it was written on. This kind of tightly coupled architecture is opposite of desired loosely coupled in the cloud as the loosely coupled architecture is one enabler for application to be truly elastic. Legacy applications tend to be stateful, and stateless applications suit better for the cloud than stateful ones. The work needed to refactoring the application completely from stateful to stateless can be unmanageable. These two reasons are likely to lead to a disappointment in the company as the end result does not bring all the benefits of the cloud. [24]

Another setback might be caused by media, when they are touting successful stories of companies that have moved to cloud or started on cloud, which have created enormous savings and success for the company. When the company tries itself to adapt to the cloud and not succeed as well as the others they heard of, it will probably cause disappointment. Not so successful stories rarely get to the front page of media because companies might not want to shout out their failures on public.

Setting expectations should not be based on success of others but the business case in support of the cloud initiative. So the key elements are designing architecture, planning and having realistic expectations of the cloudification [24].

Security is one challenge in clouds as cloud is quite new technology and there are no persons with years of experience to implement security on cloud environment. Another downfall is to think that the cloud provider will implement and handle the security of the cloud, even though the security has to be implemented also to the application itself. Cloud enables new ways to exploit vulnerabilities and to hack to systems. [24]

Even though VMs should be separated from each other, there have been cases [20] where tenants that share the same physical resource have caused a situation where a VM have started to consume compute, storage or I/O resources more than it should have. This has led to situation where other tenants have “starved” out of resources on the host machine. This kind of VM that affects negatively to the other VMs on the host is referred as “noisy-neighbor”. This can be malicious or unintentional and is a new security challenge for companies that new technology has created. Some researchers have been able to pinpoint physical server on Amazon Elastic Compute Cloud (EC2) and then extract small amounts of data from other programs on that server. [20]

3.4 Example cases

In 2011 in Germany, one company had moved to cloud computing and had their email and online documents from SaaS provider. CSP’s payment system stopped the company’s access to the emails and online documents for two days without any warning, because of an error in the payment. The incident lasted long as the CSP’s regional office in Dublin could not be reached and the emails to the CSP did not solve the problem. This works as a warning: Cloud provider’s accounting or customer management systems could be in error, so it is good to have a disaster recovery plan and knowledge how to contact the CSP in case of problem or disaster, because many cloud services rely on self-service interfaces. [30]

A good example of moving from on-premises to the cloud is Netflix. In 2009, Netflix was run from its own datacentre, but in the end of 2010 most of the customer traffic was shifted to Amazon’s public cloud. Netflix claimed in 2012 that almost 29 % of all internet traffic in North America went through their services. For Netflix predicting data traffic was a challenge. To overcome this problem they decided to take advantage of cloud’s on-demand resources and concentrate building auto-scaling capabilities to their application. [24]

In the beginning, Netflix was a large monolith Java application in tomcat container. They used cloud migration as an opportunity to re-architect system to service oriented architecture, with hundreds of individual services. This also gave to the developer teams the possibility to develop and deploy their services in their own pace. [37]

Instagram is also a good example of successful cloud user. Instagram started on 2010 and on the first day they had 25000 registered users. One million users were achieved in three months. Instagram started in the cloud from the beginning and with this kind of expansion of users they could have not been able to scale with just buying physical HW. [24]

4. ETSI NFV

ETSI is the European Telecommunications Standards Institute, whose aim is to produce globally-applicable standards for Information and Communications Technologies, including fixed, mobile, radio, converged, broadcast and internet technologies. ETSI is officially recognized by the European Union as European Standards Organization. NFV part of the name is the abbreviation of Network Functions Virtualisation. This virtualisation research work is done by Network Functions Virtualisation Industry Specification Group (NFV ISG) under the auspices of the ETSI. [15][13]

The objective of NFV ISG is not to produce standards, rather to achieve industry consensus on business and technical requirements for NFV, and the common way to meet these requirements. [13]

The issues on current telecoms networks that ISG NFV is addressing are the increasing variety of proprietary hardware and a launch of a new service or a network configuration demands installation of even more dedicated equipment. These hardware-based appliances reach their end of life earlier when the innovation cycles continue to accelerate, and this is not an optimal way to respond to the dynamic needs of the traffic and services. The ways to solve these issues are Software Defined Networking (SDN) and NFV. [14]

Major focus of ETSI-NFV is to enable and exploit the dynamic construction and management of network function graphs or sets, in contrast to the current static way of combining network functions, which can be described with NF Forwarding Graph. This forwarding graphs purpose is to describe traffic flow between network functions with a graph of logical links connecting NF nodes. [29]

The basic idea of this transformation towards virtualised functionality is visualized in Figure 7, where the aim is to transform single use hardwired boxes to virtual appliances. This is achieved by evolving standard IT virtualisation technology to consolidate network equipment types onto industry standard high volume servers, switches and storages. ETSI-NFV target is to implement these network functions in software, which enables usage of standard high volume hardware. When the network function is implemented as software, it can be dynamically moved or installed to the required location faster because there is no need to install new physical hardware. This is a key enabler for dynamic cloud based network functions, where functionality can be implemented in more appropriate places like customers’ premises, network exchange points, central offices, datacentres, etc. [29]

The technology advancements do not make the hardware useless and new technology can be implemented on the network faster. This decrease development cycles of a new network technology, because creating a new hardware to implement new network functionality takes much longer than creating new software since the verification phase does not require manufacturing a new physical hardware. Also, the delivery of new network function is faster without manufacturing and delivering the dedicated HW for the network functionality. Currently every vendor has its own implementations and bundled sets of Network Appliances. [13]

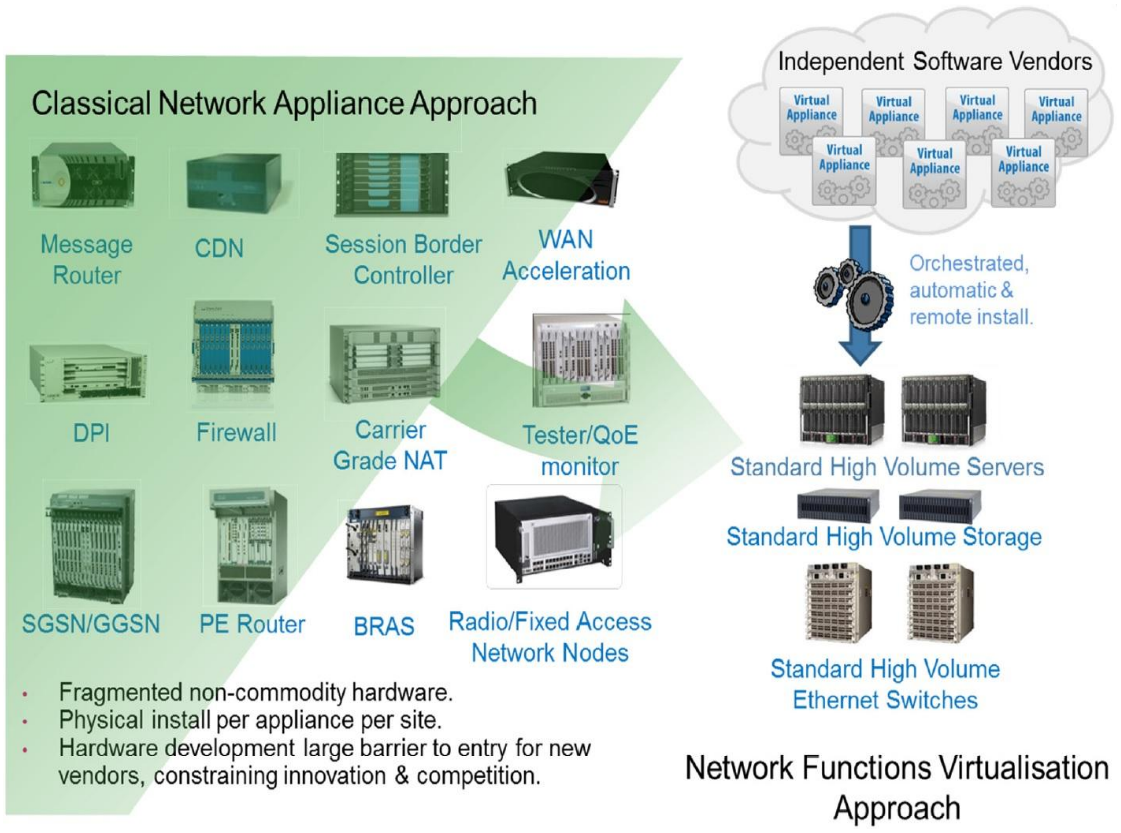
Classical Network Appliance Approach
Message Router
CDN
Session Border Controller
WAN Acceleration
DPI
Firewall
Carrier Grade NAT
Tester/QoE monitor
SGSN/GGSN
PE Router
BRAS
Radio/Fixed Access Network Nodes
Fragmented non-commodity hardware.
Physical install per appliance per site.
Hardware development large barrier to entry for new vendors, constraining innovation & competition.
Independent Software Vendors
Virtual Appliance
Orchestrated, automatic & remote install.
Standard High Volume Servers
Standard High Volume Storage
Standard High Volume Ethernet Switches
Network Functions Virtualisation Approach

Figure 7: Visualization of network functions transform to ETSI-NFV [13].

Even though ETSI-NFV can be implemented without SDN, combined they can improve the overall implementation [12]. In Figure 8 the relations of the Open Innovation, SDN and ETSI-NFV are visualized with their main benefits.

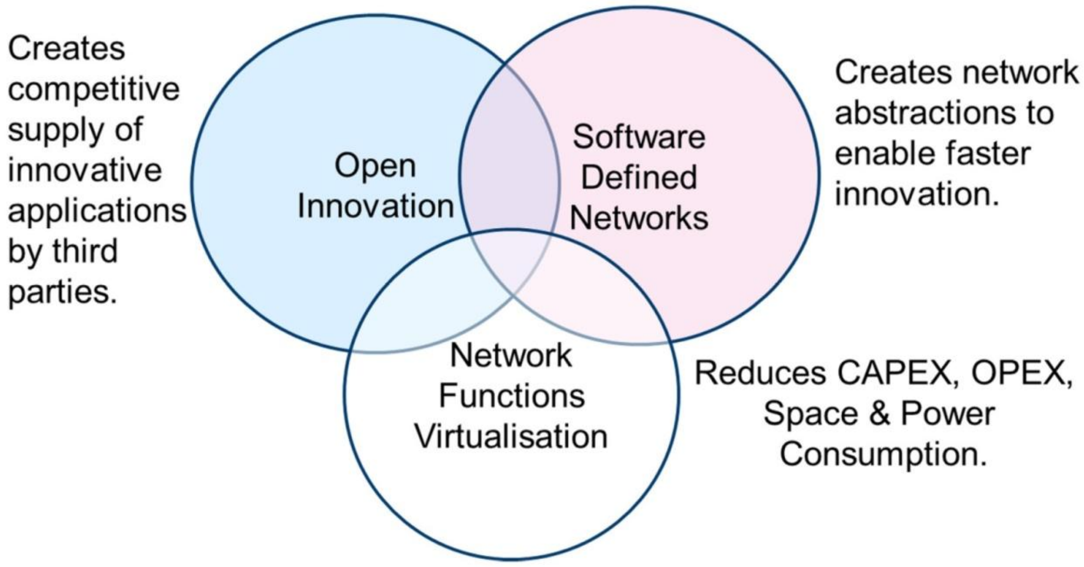
Creates competitive supply of innovative applications by third parties.
Open Innovation
Software Defined Networks
Creates network abstractions to enable faster innovation.
Network Functions Virtualisation
Reduces CAPEX, OPEX, Space & Power Consumption.

Figure 8: Visualisation of SDN, ETSI-NFV and Open Innovation [12].

SDN software can be run on the infrastructure provided by ETSI-NFV. Both have the objective to use commodity servers and switches. The use of SDN's approach of separating the control and data forwarding planes with ETSI-NFV can enhance performance, simplify compatibility with existing deployments and facilitate operations and maintenance procedures. [12]

4.1 ETSI NFV architecture

ETSI-NFV is applicable to any data plane packet processing and control plane function in fixed networks and in mobile networks [12]. ETSI-NFV Infrastructure consist of all hardware and software components that creates the environment for VNFs to be deployed, managed and executed [29].

ETSI NFV ISG has proposed an architectural framework as shown in Figure 9 [13]. By following building blocks displayed in the figure, service providers are capable of producing "NFV compatible products". The idea of this framework is to enable components from different vendors to be able to work together via defined reference points. This also clearly decouples entities to promote an open and innovative ETSI-NFV ecosystem. [13]

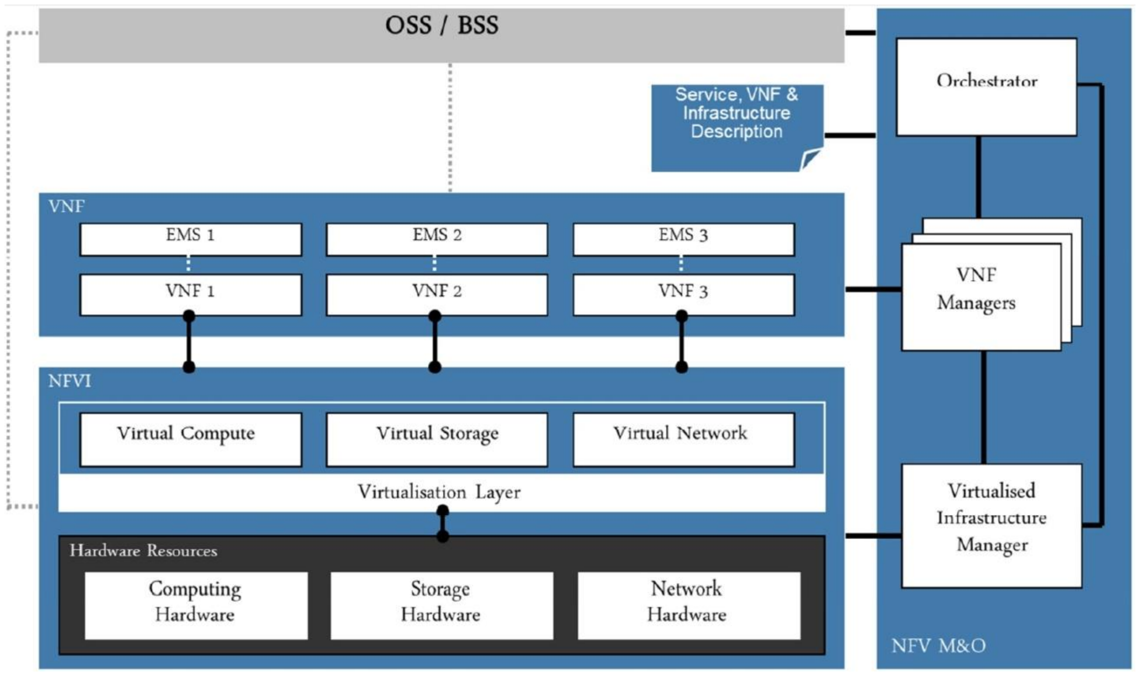

Figure 9: ETSI-NFV Architectural framework [13].

The Figure 9 is divided in three main parts: the Virtualised Network Functions (VNF), the Network Functions Virtualisation Infrastructure (NFVI) and NFV Management and Orchestration (NFV M&O).

Virtualised Network Functions are the software implementation of network functions as explained previously. In this architecture they can be accompanied by an Element Management System (EMS), if it is capable of managing and understanding VNF and its functionality. One EMS can manage one or several VNFs [29].

Network Functions Virtualisation Infrastructure consist of physical hardware and virtualised resources created from the physical ones. This virtualisation layer can be done with hypervisor as mentioned previously in chapter two. HW resources are meant to be COTS HW and accelerator components if needed. [13]

NFV Management and Orchestration is in charge of lifecycle management of VNFs and physical and/or software resources on the NFV. The NFV M&O communicates with OSS/BSS system, which is an external system to this NFV area. OSS is operations support system and BSS is business support system. OSS/BSS integrates NFV to already existing network-wide management system. NFV M&O is provided with metadata of Service, VNFs and Infrastructure requirements and descriptions, and with these information M&O is able do its tasks. [13] OSS is used more for operating the network and BSS is used for customer related functions like billing.

Virtualised Infrastructure Manager (VIM) consists of the functionalities to control and manage the interaction of a VNF with its physical and virtual resources for example by means of hypervisors. VIM also collects fault and capacity information. VIM has operations for root cause analysis of performance issues and visibility and management of the NFV infrastructure. Multiple instances of Virtual Infrastructure Manager can be deployed to operate bigger cloud infrastructure. [29]

4.2 VNF manager

A VNF Manager takes care of VNF lifecycle management which includes instantiation, update, query, scaling and termination. One VNF Manager may take care of one or multiple VNFs so there is a possibility to deploy multiple VNF Managers. [29]

Management difficulties may arise if VNF is composed of other VNFs and also if VNF is decomposed out of other VNF. This creates a situation where management interfaces may not be visible or where more management interfaces are created. [29]

4.3 Challenges of ETSI-NFV

ISG have identified challenges in the implementation of the Network Functions Virtualisation, but they have also identified possibilities on how these challenges could be progressed. Some of the challenges are described in this chapter.

Portability and interoperability challenge lies in loading and executing appliances in different standardized datacentre environments. The challenge is in decoupling SW instances and the HW beneath by creating a clear definition for a unified interface. This is important for creating different ecosystems for virtual appliance vendors and datacentre vendors and giving to the operator the freedom to optimise the location and resources of the virtual appliances without constraints. [12]

ISG acknowledges that there is trade-off with performance when using standard hardware instead of proprietary hardware like accelerator engines. The challenge will be trying to keep the performance drop as small as possible. [12]

Migration and co-existence with legacy systems is one of the challenges. ETSI-NFV implementations have to be compatible with operators’ current network equipment and different management systems. This would be a hybrid network with physical and virtual network appliances. [12]

Automation is one of the key enabler to make ETSI-NFV to scale. Implementing automation to all of the functions is crucial. [12]

Security, resilience and availability need to be implemented and assured so that the network operators would move to ETSI-NFV implementations. The cloud itself improves resiliency and availability with its on-demand character as VNF can be launched automatically when needed, but all of the components, hypervisor included, need to be secured and possibly security certified. [12]

Networks have to be stable with all different possible implementations. Stability cannot be impacted when managing and orchestrating a large number of virtual appliances between different HW vendors and hypervisors, especially when the VM is moved around in the cloud. [12]

Integration of multiple virtual appliances onto industry standard high volume servers and hypervisors, with the capability to mix and match from different vendors without causing significant integrations costs and lock-in can be challenging. [12]

NFV ETSI ISG also states that decoupling VNF from the hardware creates new management challenges. These challenges are end-to-end service to end-to-end NFV network mapping, instantiating VNFs at appropriate locations to realize the intended service, allocating and scaling the hardware resources to the VNFs, and keeping track of VNF instances' location. [29]

5. REAL-TIME NETWORK MONITORING PRODUCT

Telecommunication networks have became large and complicated because of rapid development of telecommunications technologies. This has created the need of various tools to manage and analyze networks. In highly competitive markets, analyzing the service quality, the impact of problems, and solving the problems has become increasingly important. This real time network monitoring product is created for CSPs, which are the customers of this product, to help with these problems. Development of the monitoring product was started around 1996. [40]

This product collects events and data from the network elements, and stores these data to a database for further use in troubleshooting and customer care. The data of the events can be a record of a call attempt, SMS delivery and data session. These data can be illustrated with real-time graphs with the monitoring product. The monitoring product helps an operator to monitor the quality of service in the network, the quality of the network itself, and the impact of the quality on the subscribers. With collected data, the operator can optimize the network by pinpointing the bottlenecks in real time, and after the network configurations have been changed, the effects can be immediately seen and verified. [40]

Currently the monitoring product is deployed to customer premises as physical server with the product SW. This delivery takes time because the physical servers have to be delivered to the customer premises and then a subject matter expert will go to install and configure the monitoring product according to the environment and functionalities bought by the CSP.

5.1 Architecture and operations

The monitoring product has two main parts; the network element server and the Collection server. These two parts are illustrated in Figure 10 deployed in different locations. Element servers are usually next to the network element because the data rates are high. The Element server collects, processes and stores data from the NE to a database. The Element server has a Directly Attached Storage (DAS) in order to handle the high disk IO load. Element servers’ use of database is highly I/O intensive because of high data rates that comes from the NE. Collection server can make queries to Element servers to retrieve data for further investigation. This division of roles has been made to decrease excess usage of connection resources for network monitoring so there is more bandwidth for the mobile end-user data to be transported over CSP network.

Element server can be deployed without database if the CSP only need real time monitoring and does not need to investigate older data. The sizes of the databases depend on what CSP wants to store and for how long.

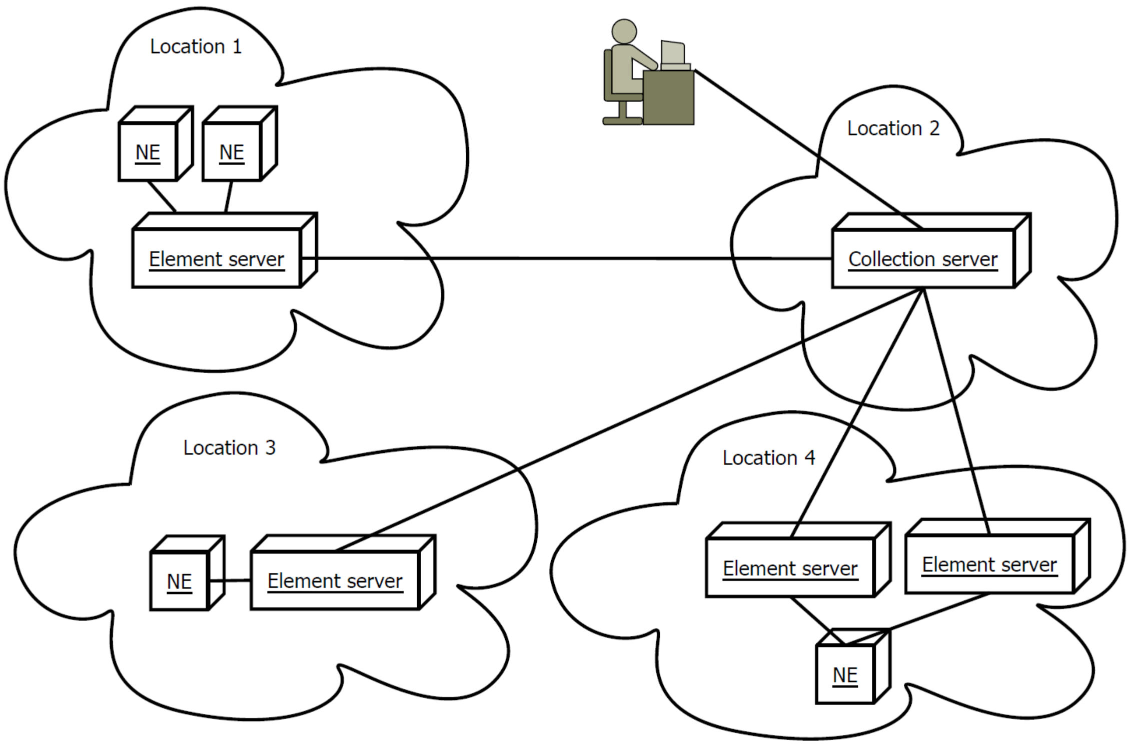

Figure 10: Possible location distribution of the monitoring product.

As seen in Figure 10, the network monitoring engineer is connected only to the Collection server and this server requests data from the Element servers based on the configuration of the monitoring product. There can be multiple users and multiple Element servers connected to the Collection server, and multiple network elements per Element server. The amount of possible connections depends on the network element technology and the maximum possible load of the network. Usually the CSP has only one Collection server, which can be divided in functional parts on their own servers to handle bigger load. The monitored network elements are 3GPP standardized components. Figure 11 shows an example of how some of the NEs monitored by the monitoring product map into 3GPP context.

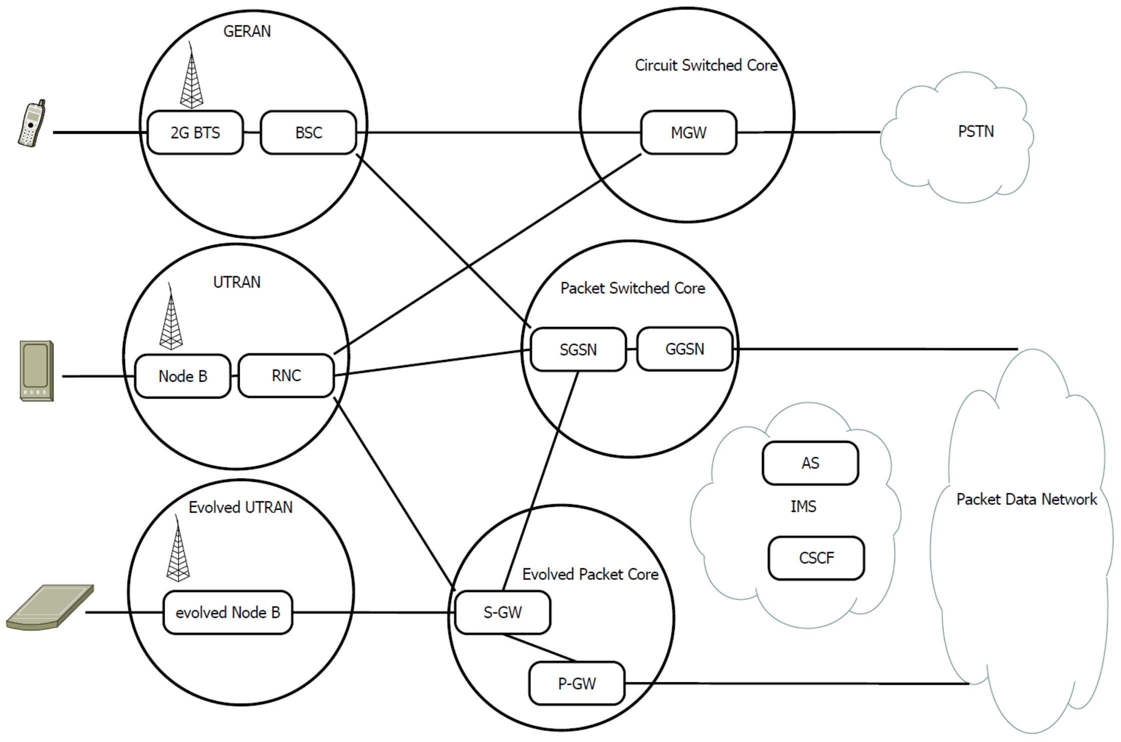

Figure 11: Few NEs in 3GPP context.

In Figure 9, the ETSI NFV architectural framework, the monitoring product is part of OSS/BSS as it supports CSP to manage networks and it does not implement any network function. Even though the Element server is next to the VNF, it is not managing the network element like the Element management system (EMS) is doing, but it fits well in the ETSI description of EMS as it is tightly related to the VNF and is managed by VNF Manager. This is the reason why the monitoring product could be considered as part of the NFV, as the functionality is tightly related to the VNF. Collection server is clearly part of OSS/BSS part as CSP usually has one network wide Collection server in their networks.

5.2 Cloud characteristics related to the monitoring product

It is already good that the monitoring product is divided in two functional parts as a Collection server and Element server. This division enables geolocational differences between these servers. Servers do not share any disks or data storages, which within the cloud is good so that the VMs can be deployed anywhere. Basically, distribution will be similar as the current way of having Element server next to the NE and Collection server in CSP's big datacentre somewhere in data traffic hub. The user connects always to the Collection server and the Collection server requests data from the Element servers, which enables the possibility of having varied amount of Element servers.

The monitoring product has the possibility for some scalability. Collection server can handle a large amount of Element servers so the amount of Element servers can be changed according to the size of the network and amount of the NEs. The Collection server is possible to be divided in parts in order to to support bigger amount of concurrent users and load coming from the Element servers.

5.3 Evaluation of best practices on the monitoring product

Comparing the architecture of the monitoring product to the best practices for a product in the cloud mentioned earlier, there are some good decisions made during the evolution of the monitoring product even though the cloud was not yet invented. Here is some evaluation of those best practices on the monitoring product.

Security has been a high priority in the product, which has lead to many security enhancements on the monitoring product, but also on the third party components when there have been evaluation whether to use them or not. Security is strictly required by the CSPs and the product has been security audited by third party security companies. These intensive audits have granted security certificate for the monitoring product. Some of the CSPs require specific security certificate or certificates for products before they make contracts. The security of the monitoring product and data processed by it is guaranteed if the monitoring product is configured and installed according to manuals given with the product. These manuals also instruct settings for the environment and OS, and are expected to be followed to ensure security. Within the cloud, the CSP is responsible to be sure not to break any laws, as the CSP might be international company and have cloud infrastructures in different countries. This can cause a situation where the cloud management moves or launches an instance of the monitoring product on the wrong side of the border. This means that the CSP has to have separation of NFVs for each country that it operates in. The monitoring product has comprehensive authorization functionality and embedded logging functionality that both restricts and records every access to the data, so that misusage of the data and information can be tracked down and further actions can be taken.

The cloud creates a unique situation where the VMs and the applications and data inside of them can be copied easily with cloud management programs. This creates the possibility to make a copy of the monitoring product and take it to a private environment and then access to the database where all of the user data is stored. Accessing to the data would be recorded only in that environment. Currently the monitoring product has all the communication encrypted to avoid man-in-the-middle attacks to collect information, but also all the databases are encrypted so that a copy of the database cannot be read without the monitoring product. This still leaves the possibility to copy the whole monitoring product in the cloud, but the security risk can be decreased with separate access rights on different persons within the CSP. A person that has access to the environment should not have access to the monitoring product and vice versa. Completely automated management of the cloud could decrease also the problem so that no person would be capable of making own copies of the VMs.

Privacy and security concerns can arise in a situation where the CSP has outsourced management and operation of private cloud to a third party. If this third party company has the headquarters in another country, the government officials in that country might demand access to all the data in their clouds by law. This can enable governmental and industrial espionage. Also, probably the cloud providing company cannot be sued because they were forced by law enforcers.

Scaling can be done currently by adding more Element servers according to the added network elements with some limitations or scaling up by changing the physical server to a better one or dividing Collection server in parts to support a bigger load. All of these scaling have to be done currently manually. Currently, some high performance capabilities are tied to specific HW, which is not good for a cloud environment since it should only use COTS HW.

The monitoring product's Element server is tightly coupled to the monitored NE. The IP addresses are statically configured for both, Element server and NE. After the connection has been made, the NE sends data which Element server receives and processes. And as mentioned, tightly coupled architecture decreases the possibility and the benefits of a dynamic cloud environment. Tightly coupled architecture does not allow restarting the component without causing malfunction or loss of data. In case the physical server would break, the instance on the server and the data coming to the instance would be lost until a new instance would be ready to replace the lost instance, if the instance is not duplicated on another physical server for redundancy. Tightly coupled architecture can be changed to more loosely coupled architecture by adding data queues or buffers between components as mentioned earlier. This would help to implement elasticity and decrease the lost of data due to hardware failure. The downside of dividing functionality to multiple servers and adding more servers would decrease the availability, because in the cloud the HW is expected to break down eventually and with more physical servers the probability of server breaking is higher.

Interoperability with different cloud implementations is now good to work with from the beginning of the cloudification. A good point at the moment is that no cloud specific components or functionalities are used on the monitoring product. The monitoring product functionality is tied to the Windows® OS so that if a cloud supports Windows® OS then there should not be reason to expect that the monitoring product would not work on that cloud environment. Performance might be different between clouds because the supported hypervisors affect to the performance of the virtual components. Other cloud environments might require own Windows® OS image creation.

When preparing for disasters, the monitoring product requires redundant VM as the product is stateful, which is not the best option for a cloud environment. But for this kind of product it is almost impossible to be stateless since some of the functionalities rely on previously received data and on a change in the data. The single point of failure in the monitoring product is the Collection server, but it can be divided in few parts on different servers, which in case of failure will make only part of the functionality unavailable for a certain period of time. In the Element servers the most critical point is the database where all data from the NE is stored. These databases are protected against disk failure by using RAID.

The performance of this monitoring product is good and the bottleneck has been the Element server, but it has been designed to be able to be divided in parts to support bigger load from the NE. The biggest bottleneck at the moment is the use of databases: what data is stored and for how long. The demand periods of the monitoring product follows the users' demand periods in the network so it increases the maximum capacity requirement of the cloud.

Efficiency is shown in the monitoring product so that not every possible component needs to be installed. On installation phase only the required NE technology support is installed to each Element server and Collection server. Afterwards, additional support for other NEs is possible to install. Full blown installation versus lightweight installation has great difference in functionality and requirements. The monitoring product requires only computing power to handle real-time monitoring functionality. Depending on how much and how old data the CSP wants to store for troubleshooting, the extra disk space is required for the databases and the high IO requirement for the storage also applies.

Divide and conquer has been implemented in some state in the Collection server as it can be divided in parts on their own servers. Load balancing is not yet implemented to the monitoring product, but the Queue system proposed has load balancing functionality between Element server and network element.

Element server placement close to the network element is according to the stay close from OpenStack guideline. This division has been done due to high bandwidth requirement between Element server and network element.

5.4 Possible improvements

Currently, all the components in the monitoring product have partly their own management interfaces, so the access to the product can be done from different components for different management and configuration purposes. This kind of decentralized management and configurations makes it harder to manage and configure the monitoring product and it makes it harder to also automate the management and configuration. Transferring the configuration information to the Collection server from where the Element servers would get their configurations would require more intelligence in the Collection server in order to be able to determine the purpose of each Element server and to give right configuration for each of them.

The Collection server includes licensing management so when a new Element server is launched it will register itself to the Collection server. This way the licence in the Collection server can limit the amount of Element servers. The launch of a new Element server would need to be initiated by the Collection server, so that the application management would not try to push new Element servers when the Element servers would be experiencing high load and the licence would not allow more Element servers. This would require a solution to a situation where the Element server is lost due to HW failure or failure in the VM so that the registration is freed from the Collection server. The application management entity in the cloud would be probably the best place to have the information of the IPs of the components of the monitoring product so that it could give the IPs of the components where the new Element server would need to connect and where it should physically reside.

Queues could be implemented between network element and Element server, and also between Element server and Collection server as seen in Figure 12. If all of these components would reside on separate physical servers like in the figure, the loss of data due to one hardware failure would be very low. If Element server would be lost, only the data on that Element server would be lost and not the incoming data as it would be buffered in the Queue 1. Queue system would help to make the system elastic if the Element servers would be capable of horizontal scalability, because then the queue system could divide the data flow from Queue 1 to the Element servers according to their capability and Queue 2 could send the queries from Collection server to all proper Element servers. New Element Server would be easier to add to the system because the information of the Element servers would only need to be on the queues on both side of the Element server, if the licensing would not limit the amount of Element servers. If the licences would be handled by the Collection server then the previously mentioned self registering of the Element server would be done through Queue 2.

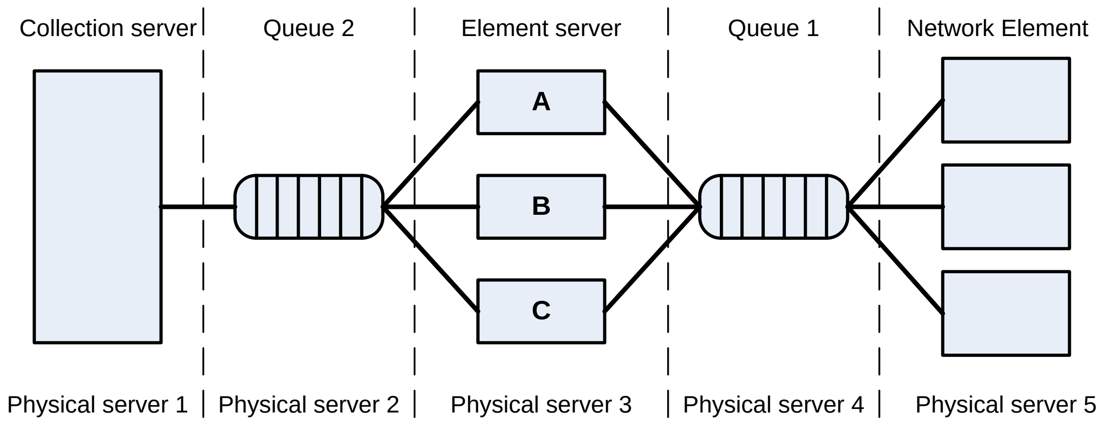

Figure 12: Queue system for the monitoring product on scale out situation.

If there would be more Collection servers, Queue 2 would require intelligence to know from which Collection server the query to the Element servers came to be able to forward the right response to the right Collection server. Even with a system like in Figure

12, the Queue 1 would need some kind of functionality to detect and inform Element server if one of the network elements would disappear without graceful shutdown and also with graceful shutdown like in scale-in situation. Currently the monitoring product creates an alarm if the connection to the network element is lost. Implementing functionality according to Figure 12 would require a new functionality in the Element server in order to understand weather network element is lost or Queue 1 is lost and create alarm according to it. Detecting unexpected lost of network element can be handled with heartbeat functionality between Queue 1 and the network element, but for expected graceful shutdown the network element would need to send a message about the shutdown to Queue 1 or the cloud manager program would need to send information to Queue 1 or straight to the Element server about the shutdown.

Implementing horizontal scaling to the Element server is difficult because of the use of databases. As an example to the Figure 12, if Element server C would be dropped due to scale in, the database attached to that Element server would still require to be accessible for the queries coming from Collection server. This could be solved by implementing only one database so it could be used by all the Element servers of the specific network element. If two databases would be required because of the amount of data coming from the network element, the minimum scale in point could be two Element servers that would be always present to handle the databases. So the minimum amount of Element servers is directly related the minimum amount of required databases for the peak time. When doing the evaluation of the maximum writing capability and requirements for the database, there need to be considered also that during the highest peak of traffic there can be also queries to the database, which can effect to the maximum writing capacity of the database.

Another challenge to make Element server to scale horizontally is the SW. Element server has been developed for almost two decades with a mindset of having the computation on one physical server in order to keep it efficient. This means that it does not have dynamic capability to share functionality or load with another Element server.

Implementing Queue system according to the Figure 12 and in relation to Figure 10 it would look like in the Figure 13. The final solution and deployment will depend on the CSP and on the elements the CSP wants to monitor.

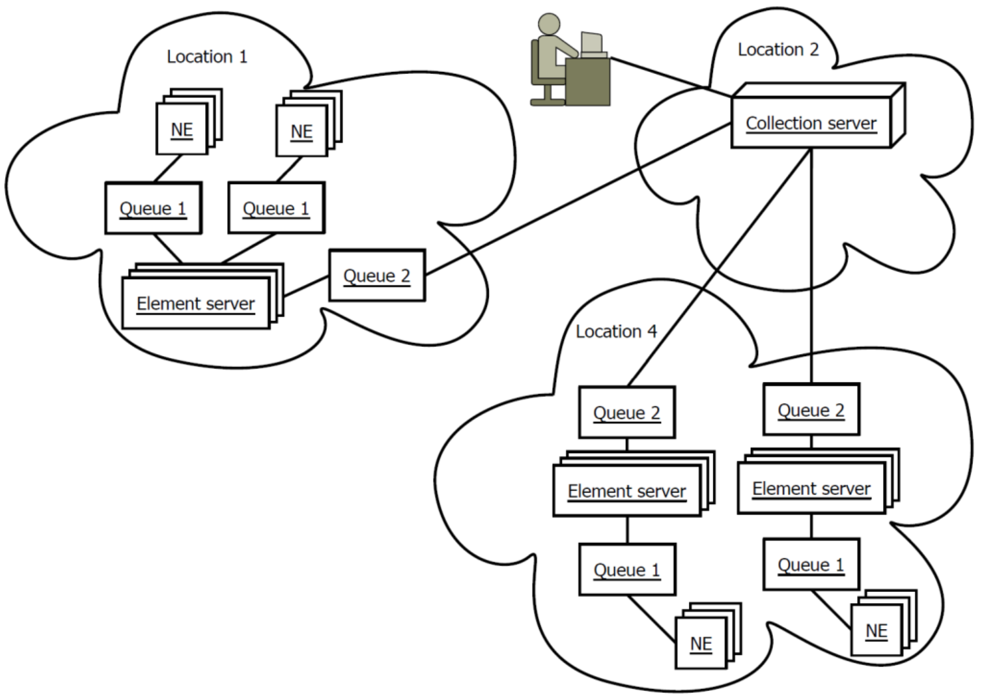

Figure 13: The monitoring product with queues and scaling.

As it was mentioned previously, programs need to have all functionalities and configurations possible to be done through an API or similar and not only with human interaction through GUI so that another program is capable of controlling and configuring the program. This is also needed for the application to benefit from automation and dynamic configuration without human interference in the cloud. This way the cloud manager can set necessary settings to the program so that the program is capable of self discovery of the role and rest of the settings, according to the environment so that it can fulfill its purpose in the cloud. API is the preferable way, but it can be done with scripts or configuration files depending on the chosen cloud management program capabilities.

The program should work without human interference after the configuration files, which describe the managed system, have been created for the cloud manager and the system is set according to the files. Scaling by adding or removing VMs is done automatically by the cloud manager and it has to have access to all VMs in the environment that it controls in order to be able to reconfigure all needed VM when something triggers the scaling in the cloud environment. When going further with this, in order to save computing resources, it would be better to separate the GUI from the program so that if there is need for human interaction, it can be done remotely from another computer or VM. Without GUI, it is possible to save computing resources by installing Windows® server OS as core version without GUI and then the application without GUI as a service. A similar possibility in Linux OSs is possible. According to Microsoft [42] the Server Core installation requires approximately 4 GB less space than a Server with GUI installation on Windows® Server 2012 editions.

Porting the monitoring product from Windows® OS to Linux based OS is a desired property, but because the code has lots of Windows® specific enhancements and components, it will be difficult to accomplish. It will require a deep inspection to the code base of approximate of 500 000 lines of C++ code. Therefore, to move the monitoring product to work on Linux will take time. The best option could be to make the monitoring product's code to be able to build so that it would also work on Linux. Linux OS is currently desired by the CSPs for cost savings and licensing issues compared to Microsoft OSs. For the vendor it would be better to support only one platform because testing for two completely different OSes will take more resources and time.

6. CLOUDIFICATION OF THE PRODUCT

This chapter describes the used environments and applications needed for the cloudification of the monitoring product. The workflow of the image creation is described in detailed level and possible improvements to the monitoring product are described in higher level. Cloudification started from the point where the monitoring product is working in virtualised environment but with different products involved.

During the virtualisation phase of the monitoring product, arose performance limit with the size of VM. Scaling up to have bigger VM would not increase the performance of the product. The limitation comes from the underlying CPUs and RAM. Processor affinity problem also arose when the VM had processors from different sockets, and it did not perform well because the threads were changing the socket where they were computed. The non-uniform memory access helps the monitoring product to perform better as the processors access to the closest memory so the NUMA setting is the preferred setting for the monitoring product's environment.

Because of the cloud characteristics of using COTS hardware on the physical layer, it is quite impossible to give capacity statement for the monitoring product. The performance of the HW, used hypervisor, and the connections between physical servers affect the performance of the monitoring product. The size of physical server limits the maximum size of VM, which could take all resources from a single server. The use of databases will affect also the performance: how the disks are attached to the cloud and what kinds of disks are used. OpenStack allows VM to be run on desired host server or server group, and this makes it possible to run Element server on a specific host server that would have a database disk array attached straight to it or it would have the best access to the disks.

The Element server is very disk intensive and needs enough IOPS for the database. Disk latency and disk capacity affects also to the performance of the database. In the cloud it might be difficult to use some additional disk system for database as it would need to have good enough connection to the application writing in to it. OpenStack allows users to create persistent storage volumes with the block storage called Cinder. These volumes could be used for databases and after filling one up it could be moved to an external disk storage for later use if needed.

6.1 Used cloud environment and products

In this cloudification, KVM is used as hypervisor and Red Hat Enterprise Linux® 7 (RHEL 7) as underlying OS on the host machine. KVM is not completely type-2 hypervisor nor type-1, because it converts the Kernel of the host OS as type-1 hypervisor [25]. KVM was chosen because it is free software licensed, which gives quite free hands to use and develop it. Also, many CSPs want to avoid proprietary components in order to decrease CAPEX and/or OPEX. KVM is also the best option when using OpenStack as it is fully supported [22]. OpenStack was chosen as the cloud layer because it is open-source cloud computing platform and CSPs are interested on OpenStack deployment. The used OpenStack version is Juno, and is provided as OpenStack Platform 5.0 bundle from Red Hat, Incorporated. This solution was picked because Red Hat is able to give commercial enterprise-level support. Free version of this bundle is called RDO.

The VMs created in the cloud have Windows® Server 2012 R2 OS because the monitoring product supports Windows Server OS and at the moment the 2012 R2 is the latest version of Microsoft's Server OSs.

VirtIO-win drivers are paravirtualised drivers for Windows® OS. These drivers enable direct access to the devices like in paravirtualisation and make the use of those devices faster by decreasing guest I/O latency and increasing throughput to near bare-metal levels. These are recommended for guests running I/O heavy task applications, like the monitoring product. VirtIO-win is licensed as GNU General Public Licence, version 2 (GPLv2). [35]

Cloudbase-Init is open source project that brings features to Windows® that are brought to Linux by cloud-init. Some features that it brings are: user creation, password injection and SSH public keys. Password injection only happens when the instance is launched. [7]

Answer file is Microsoft's way of automating installation of Windows® OS. It is XML-based file that contains setting definitions and values for the OS. With this file, Windows® OS can be installed completely automatically without user interference. It can include all the configurations that can be set during the manual installation as well as other configurations. Some of the configurations set with *answer file* are: security and firewall settings to enable remote desktop connection, language settings, Internet Explorer settings, and a product key. With *answer file* the user can also run scripts and commands in the Windows® OS. [38] Microsoft has free Windows® Automated Installation Kit that helps in creating and verifying *answer files*.

The System Preparation (Sysprep) is a tool to prepare an installation of Windows® for duplication, auditing and customer delivery. Sysprep removes unique information from the Windows® installation. [41] Sysprep is used to generalize the created image to be used on OpenStack.

6.2 OpenStack cloud operating system

OpenStack is an Open Source Cloud Operating System. It controls large pools of compute, storage and networking resources [33]. All of this can be managed through a dashboard called Horizon on browser or using the native OpenStack API or the Amazon Elastic Compute Cloud (EC2) compatible API or command-line interpreter (CLI) [21]. APIs are meant to automate or to build tools to manage the resources. These APIs enable the usage of a third party cloud manager as mentioned in chapter four of ETSI NFV architecture model, referred as VNF Manager. Also, the whole VNF Management and Orchestration entity can be third party applications because the APIs are open source. Figure 14 illustrates different services of OpenStack.

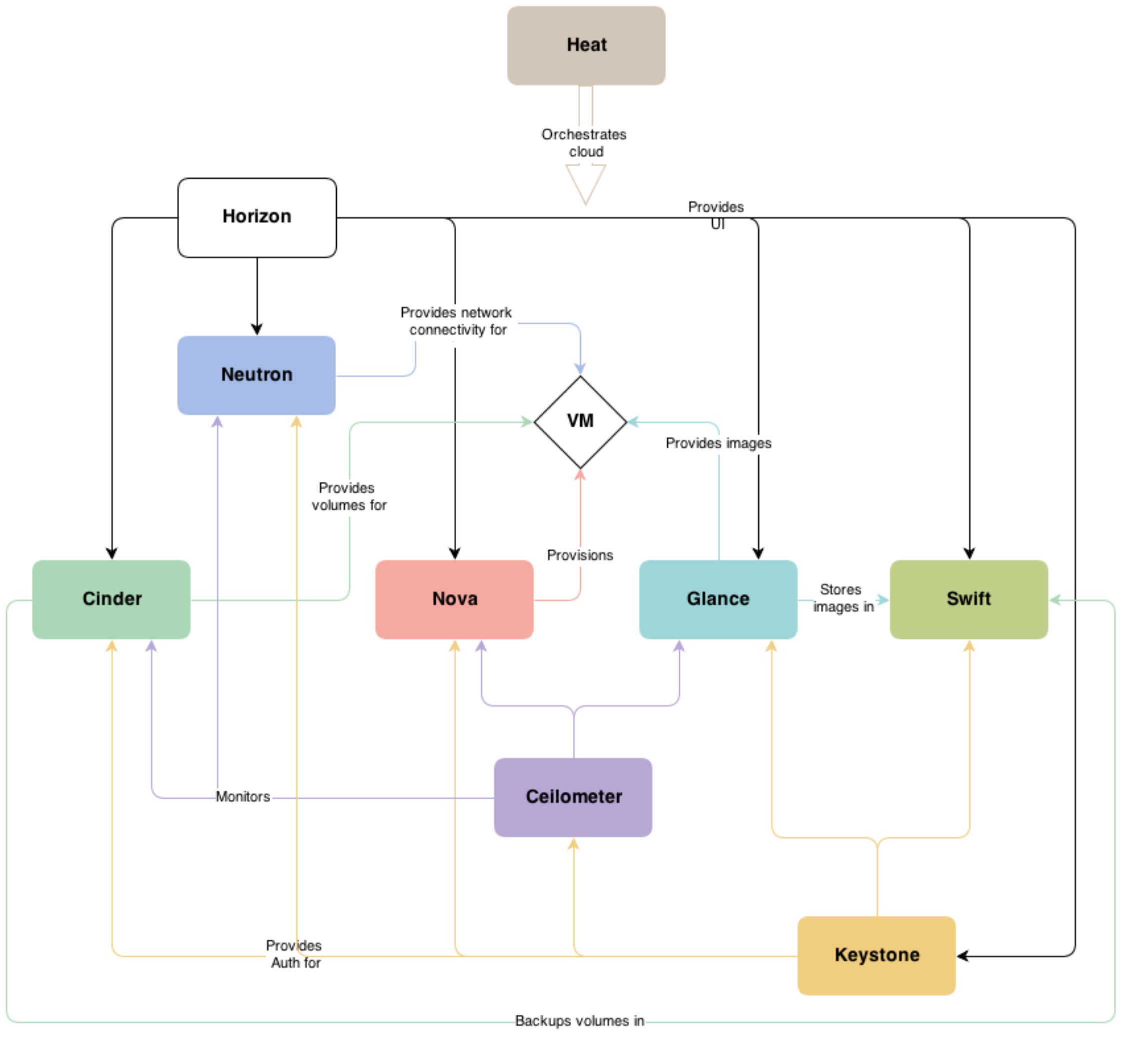

Figure 14: OpenStack conceptual architecture [10].

As seen from Figure 14, the services and their purpose are [32]:

- Nova is OpenStack Compute service and it is used to host and manage cloud computing systems.
- Keystone is for Identity service, which provides authentication and authorization.
- Neutron is for networking and it enables Network-Connectivity-as-a-Service for other OpenStack services.
- Glance is an image service, which stores and retrieves virtual machine disk images.
- Swift is an object storage and can be also used by Glance to store images.
- Cinder is a block storage, which provides persistent block storage for running instances.
- Ceilometer is for telemetry and it collects information of usage in the OpenStack cloud for billing, benchmarking, scalability and statistical purposes.
- Heat is for orchestrating multiple composite cloud applications using templates.

VM instances can be accessed with console on Dashboard on browser. Instances can be accessed also with remote access programs according to the OS on the VM. Windows® VM can be accessed outside of the cloud using Microsoft Remote Desktop if the right settings in the Windows® in the VM instance is set, floating IP address have been assigned to the VM, and OpenStack security group settings for the VM is enabling remote connection.

OpenStack offers few different default sized instances called flavors. The size of the VM and the instance refers to the amount of virtual CPUs, RAM and disk space given to the VM. Users with sufficient rights are capable of creating custom sized flavors for their environment. In OpenStack, the user can launch different sized instances from the same image. [17]

6.3 Cloudification environment infrastructure

This cloudification test environment consisted of three servers, one for creating images and two to host the OpenStack cloud itself. This environment is illustrated in Figure 15. The first steps were to set the physical servers running and to install the KVM, RHEL® 7, and OSP 5.0 to the two servers that created the infrastructure of the OpenStack. On the installation it is decided how the cloud architecture is implemented. With two node cloud test environment, it is good to divide nodes as control-node and compute-node. The compute-node is purely for running the virtual machines and the control-node handles all of the other cloud functionalities.

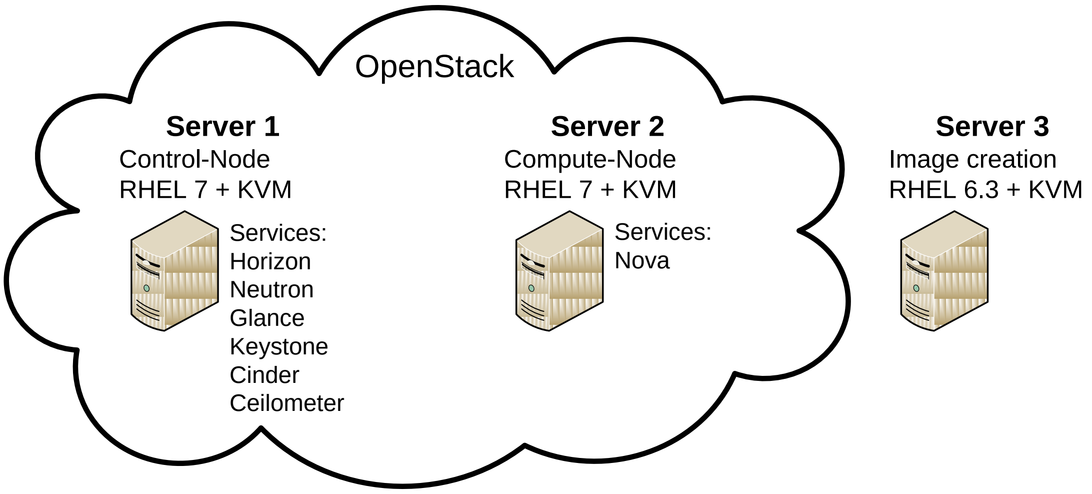

Figure 15: Test environment.

As seen from Figure 15 the OpenStack is divided in two parts. Server 1 has most of the OpenStack services and is called Controller-node. Server 2 has only Nova-compute service running on it. Most of the Nova services are on Server 1 and OpenStack System Info marks Nova on Server 1, but here in Figure 15 Nova is marked to be on server 2 because it is the server where the computing of VMs is done.

6.4 Workflow of the cloudification

Figure 16 illustrates the workflow steps from the components to create an image, to running instances in the cloud. The step one in Figure 16 is to create a cloud bootable Windows® server image. This image can be done on top of KVM using *virt-install* or GUI version *virt-manager*. To create more automated procedure, it is better to use CLI version *virt-install*. For automating Windows® installation and configuring required settings it is advisable to use *answer file* to make unattended installation. Using *answer file* for installation gives the possibility to create new bootable images faster, as different customers might have different requirements, which can be preconfigured for the image. Unwanted configurations can be cleared with Windows® *sysprep* functionality at the end of the image creation. When booting the virtual machine in cloud it will ask essential configurations that were cleared by the *sysprep*. *Sysprep* was included in Cloudbase-Init so it was run at the end of the image creation phase.

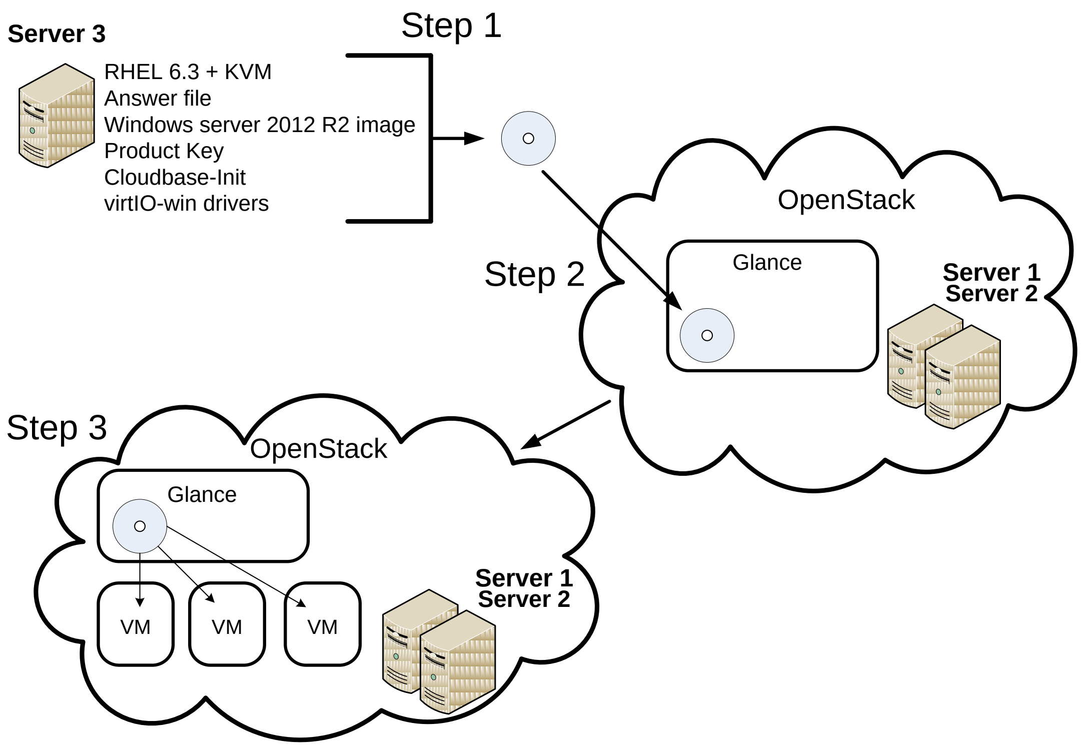

Figure 16: Illustration of image creation workflow.

In step 2, after the image is created, the image is uploaded to Glance in OpenStack. In this setup, Glance is located on Control node and it is not using Swift for storing images. After the image is uploaded to Glance, new VMs can be instantiated from that image in step 3. The first VM instantiated from the new image in compute node will take time because the image data have to be moved to the compute node where the instance will be started and run. The next VM booted from that same image will not take so much time because the compute node has cached the image data.

Creating and adding extra volumes for these Windows® VMs can be done by OpenStack CLI or through Dashboard or with APIs. To be able to use extra volume with Windows® it is required to make it firstly online on Windows® Disk Management and then initialize it to be non-default master boot record (MBR) partitioned. This is now automated by PowerShell scrip that can be executed remotely. To detach the volume it is enough to turn it offline in the Windows® Disk Management. This can be also automated but requires validation procedures so that the right volume will be detached. One possibility is to create a file with information of the created volume when adding the volume so that on the detach phase it can be verified which volume has to be detached. If the situation is that there is only one volume attached at that time, it can be verified with few PowerShell commands that it is not a system volume. After detaching the volume, it can be attached to another VM.

In this environment two VMs are launched to host the monitoring product’s components. Collection server is installed on VM 1 and Element server is installed on VM 2 as seen in Figure 17. Each VM gets its fixed IP assigned by OpenStack when the VM is launched. This IP is used for the communication between instances inside the cloud. Floating IP is assigned by the users from a given set of IPs and it is used for communication with networks outside the cloud. This remote connection is used to configure and troubleshoot problems in the product. [26]

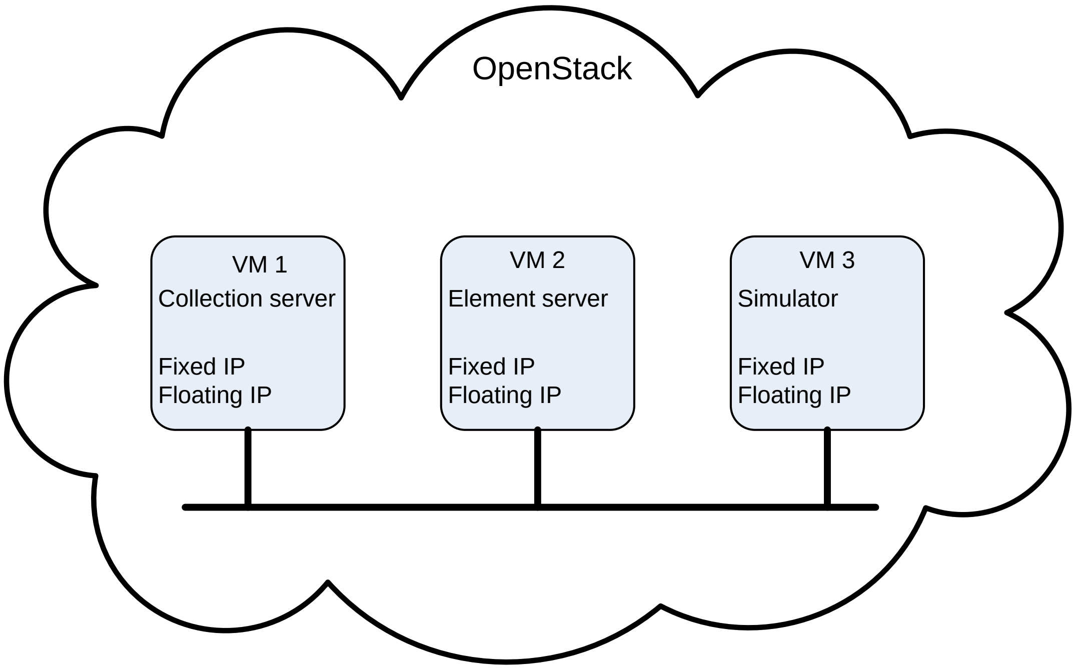

Figure 17: Test VMs.

VM 3 in Figure 17 has a simulator to generate network traffic for the Element server. After installing each component, they were configured accordingly and given the IP of the VM that they should be connected to. Data flow starts from the Simulator that generates data to Element server, which process the data and creates a report for the Collection server.

6.5 Problems faced during the cloudification

For the author, this work started with almost no knowledge of cloud computing or what it meant. Some knowledge and concept of virtualisation were acquired from studies in TUT and from the internet.

In the beginning, the target was to start working with DevStack version of OpenStack. DevStack is mainly targeted to developers to easily set up an environment from source repositories for development and operational testing of OpenStack. The starting point did not succeed because there were lots of problems on installing DevStack on the testing environment. After some time trying to get it working, it was concluded that DevStack will be not used. Later, support from RHEL® OSP expert was available, which directed the decision to use more mature and commercially supported platform.

With the help of the expert, the installation of the environment was fast and there was support available on problematic situations regarding the environment, and the main target in this thesis was not administrating and configuring the cloud.

The used Windows® based OS creates one of the challenges in the cloudification of the monitoring product because of the licensing terms of Windows®. It is challenging to sell the OS into customers cloud infrastructure without selling also HW. The cloud is a different environment from the more traditional server HW, OS, and SW delivery, because the OS is not installed on one specific HW where it would stay for its whole life cycle. Also, the licences of the third party software that are used to install Windows® on the OpenStack and KVM have to be evaluated for commercial use in order to avoid the leakage of own IPRs or the violation of a third party IPRs and licensing terms.

If the delivery to the customer consist only the software of the monitoring product, Windows® installation and configuration would be completely managed by the customer. In this kind of delivery, only a requirements document or a configuration file would be delivered with the monitoring product so that the configuration of the OS would be on the customer's responsibility. The configuration file could be the *answer file* or PowerShell script, which can be run on live instance before the installation of the monitoring product. This would also release the vendor from the licence restrictions and would give the customer the freedom of choice on how to install Windows®.

The lack of documentation for the Cloudbase-Init created difficulties to automate the installation of Windows® image with *answer file*. Using PowerShell command to install the Cloudbase-Init was not straightforward and the result file created during the installation did not inform which components were installed. The differences between options during manual installation and attributes given when installed with PowerShell command made the installation difficult. Ensuring that *sysprep* was run during the Cloudbase-Init had to be checked from the Windows® registers.

6.6 Cloudification maturity model

During this work, there was a clear indication that some qualities were important in order to be able to work in the cloud and some other characteristics are required to achieve the benefits of the cloud. Referring to a colleague with his presentation of maturity model, Figure 18 illustrates how a product matures little by little towards the nirvana state of cloudification. Of course this nirvana state might not be feasible for all existing products and it has to be considered in which cases it is possible to achieve and which cases it would be better to create a new product from the beginning. The steps where the monitoring product fits are circled with a frame in Figure 18.

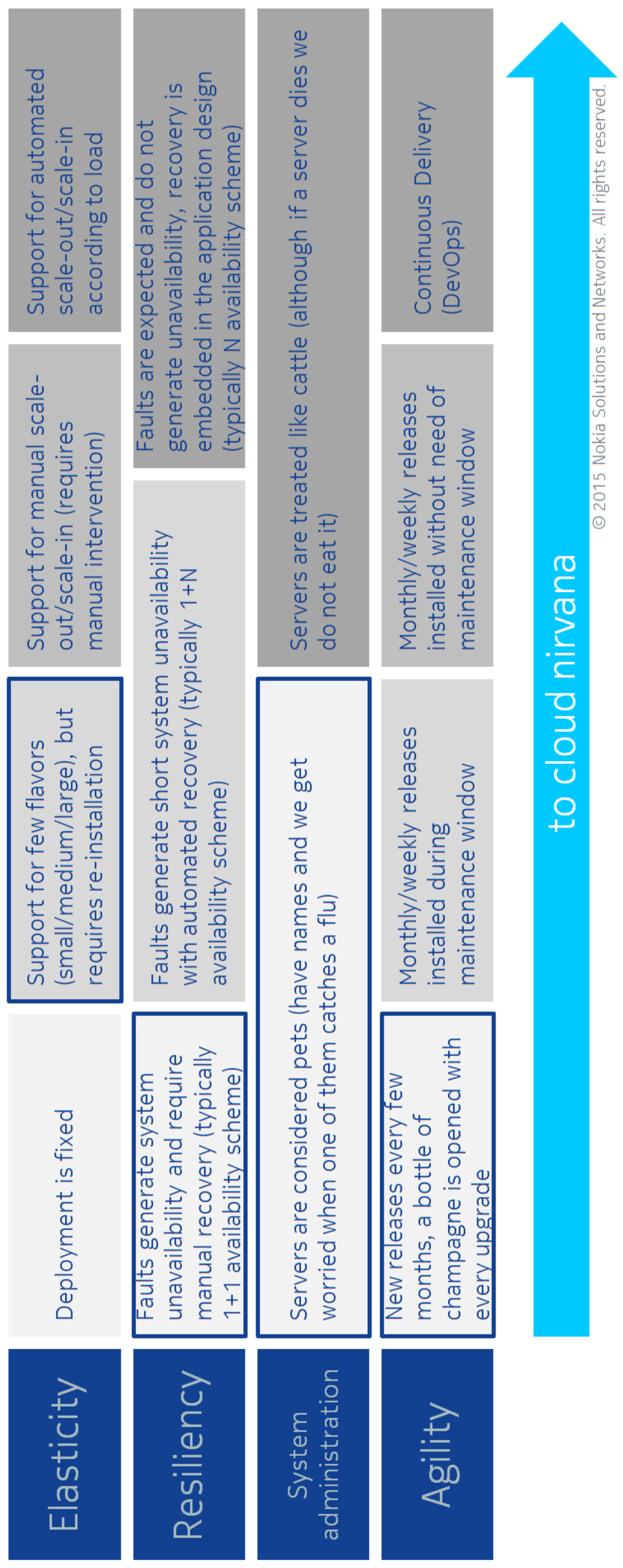
Elasticity
Deployment is fixed
Support for few flavors (small/medium/large), but requires re-installation
Support for manual scale-out/scale-in (requires manual intervention)
Support for automated scale-out/scale-in according to load
Resiliency
Faults generate system unavailability and require manual recovery (typically 1+1 availability scheme)
Faults generate short system unavailability with automated recovery (typically 1+N availability scheme)
Faults are expected and do not generate unavailability, recovery is embedded in the application design (typically N availability scheme)
System administration
Servers are considered pets (have names and we get worried when one of them catches a flu)
Servers are treated like cattle (although if a server dies we do not eat it)
Agility
New releases every few months, a bottle of champagne is opened with every upgrade
Monthly/weekly releases installed during maintenance window
Monthly/weekly releases installed without need of maintenance window
Continuous Delivery (DevOps)
to cloud nirvana
© 2015 Nokia Solutions and Networks. All rights reserved.

Figure 18: Cloud maturity model.

From the application point of view, this maturity model explains how elasticity is improved over the development. The first step in elasticity is present as the application has a fixed deployment. The second step would be taking to a use the possibility to scale up and down manually. The third step would require architectural and functional support for scaling out and in, but would still require a person to handle the scaling. In the Nirvana state the application would be capable of scaling out and in automatically triggered by the load in the environment.

At the moment the monitoring product is in fixed deployed situation, but the product can be scaled up so it can be said that it is on the second step of the model. Scaling out can be done by adding or removing Element servers according to the amount of monitored network elements, but requires lots of manual steps to link the network element to the Element server and the Element server to the Collection server, and change the configurations. Currently, if the new network element instance is launched in the cloud, it would require that the Element server would be ready to monitor it before the NE would start taking data, otherwise there would be loss of monitoring data. Also, if the NE instance is dropped due to scale in, the monitoring product would create an alarm on the system because there is not yet any interface and functionality in the monitoring product that it would get information from somewhere about that the monitored NE is going to be gracefully shutdown. Similar internal notification between Element server and Collections server is missing. These interfaces are not implemented in the monitoring product because the current way of deploying the product and network elements is fixed.

Resiliency is a sort of redundancy to have a back up, if some component fails. In the first step of resiliency in Figure 18, every component has one backup component. If the active component becomes unavailable due to HW failure, then the backup component takes its place. But the backup component requires manual configuration to be able to work properly. Because the configuration is required, the system will be unavailable and the manual configuration makes the unavailability time longer. On the second step towards nirvana, the recovery will be automated and the unavailability time will be shorter due to the automated actions. In the nirvana state the faults are expected and will not make the system unavailable because of the active backup components and functionalities implemented in the application. This might be cumbersome to achieve because components need to know that the transmission of data have been successful or some kind of queue, as mentioned previously, need to be implemented between components. The fact that the components of the monitoring product are stateful will make resiliency nirvana state difficult to achieve because the components require active copies of the components ready on other physical server in case of HW failure. The monitoring product is on the first step on the resiliency.

System administration affection to the servers will change to more unemotional as described in the figure as pets and cattle. In the first step the server will be taken care of and maintained and fixed if needed, because it could be that the applications in the cloud are not ready for breakdown of the physical server. In the nirvana the server in the cloud will be more like cattle, when the life cycle of the server comes to the end or some component breaks in it, then it will be thrown away and new one will be put in its place. This is possible because the applications in the cloud should withstand the break of the server. As mentioned earlier, the COTS HW are expected to break down easier than a normal server HW and it might be more affordable to just change to a new one than repair the server.

Agility in the maturity model starts with the delivery of a new release every few months so that it will create short unavailability time. On the second step the releases will be installed on maintenance window weekly or monthly. On the third step the releases are installed without the need of making the application unavailable. On the nirvana state, all new releases and upgrades will be delivered after the functionality has been tested. This will enable fast reacting to bugs and security issues on the application. Achieving the nirvana state requires that the application architecture is very loosely coupled so that making changes to a component will not break other components functionality. This nirvana state is not completely in the vendors target because the private clouds of CSPs are closed environments so the deliveries cannot be daily or weekly, but still the vendor should not slowdown the CSP if they want to proceed to more frequent deliveries.

Possible ways for delivery would be sending an update package or updated virtual server image or updated installation package. The update package could be installed during a maintenance window during the night as the network load is a lot lower. And the server images for the cloud could be recreated with adding the update package. One way to deliver an update would be sending a new cloud ready image, which could be changed during a maintenance break, but would require copying the state of the old instance to the new one before directing the data to the new instance. Propably the easiest way would be sending an updated installation package, which the CSP could install on a running instance or create new cloud ready server image. The delivery model probably depends on the desire of the CSP and requires some more evaluation on security and certainty of successful upgrade delivery.

6.7 Improvements to support cloudification on the product

Time wise there were no improvements done to the monitoring product. The output is mostly on the evaluation for possibilities and pre-study for decision making and to make sure that the product works in OpenStack.

Automated image creation to the OpenStack will help in the future development. Changes to the image creation can be done to the *answerfile* or to the script that initiates the image creation. This enables easier production of the Windows® image for OpenStack with different settings and updated components.

The first step could be to improve the deployment to the cloud and going towards delivering only the software of the monitoring product. This enhance would be done so that the installation can be done automatically without GUI. This could help in two possible delivery cases; when the monitoring product is installed on a new VM instance or when the monitoring product is preinstalled on an image that is delivered to the customer.

If the monitoring product would be preinstalled on the image, it would mean that there would need to be many different images prepared to support all possible installation varieties of the product. This would happen on both Collection server and Element server installation. As stated previously, Collection server has lots of capabilities but not all of them are required to be installed and it depends on the functionality purchased by the customer. One option could be to deliver an image that would have the installation files included and, when the instance is launched, the Collection server or the Element server would be installed with the required components and settings according to a configuration file given by the Cloud manager.

Currently, the Element Server requires to be manually configured via local embedded GUI. These kind of properties need to be transferred to the Collection Server so that the centralized control of the monitoring product would be in one place. This would support easier life cycle management and elasticity. The control of the capability limits and size limit in scaling of the monitoring product would be easier when Element servers would request the configuration and authorization from the Collection server.

Currently upgrades to the monitoring product are delivered and installed manually to the server. Upgrade procedure disables the usage of the product for the time of the upgrade. The cloud way of delivering upgrades to the products is continuous and should not interfere the usage of the product. The upgrades could be delivered so that the new instance of the component is launched and that new instance is upgraded. After upgrading the new instance, it would replace the original instance and then the original one could be deleted. This would require a new functionality in the product so that it could export and import all the stored data and configurations. These steps should be also automated for easier management.

Due to lack of standards, interoperability with different clouds requires additional work. However, since at the moment there is no cloud specific implementation done except the cloud image itself, there is no vendor lock. Future interoperability consideration has to be taken into account when evaluating different cloud management product compatibilities, so that the monitoring product is easier to add to the environment that the CSP already has, which could improve sales possibility. Also, implementing performance monitoring to the monitoring product could be useful in a situation where the monitoring product could inform the cloud management product about its performance so the management can use that information on making scaling decisions.

7. CONCLUSIONS

For a product that is already on the market and evolving, it is cumbersome to apply all of the desired improvements. Cloudification is achieved step by step by implementing features which works also on current deliveries, similarly as in cloud maturity model. To have all the benefits of the cloud on fast schedule on a mature product, it would require lots of people to work on it and would mean slowdown of current feature implementation requested by customers. Currently, the target is to support multiple cloud environments to avoid vendor lock-in. This is also in line with ETSI NFV as they are driving towards cloud agnostic and interoperable way. To implement an automatic way of connecting to the monitored network element, the network element needs to have implement interface and functionality to support this dynamic connection to enable scaling out and in of the monitoring product.

Even though security is not the main point in this work, it has to be considered very thoroughly. The monitoring product is collecting and storing data which includes identification data and because of that, laws and regulations are affecting to it. This means that the monitoring product cannot be in a public cloud with all the risks of unauthorized access to data. Cloud brings a lot of new security considerations.

Supporting of deployment on bare-metal should be continued as there are still benefits of having dedicated high performance hardware installation for the monitoring product. The transition of telecommunication networks into cloud is gradual and will be done considering the existing investments, which would mean that the monitoring product system is partially in the cloud and partially on a specific hardware. A remote location could be a place where the Element server is deployed on bare-metal next to the NE to save used bandwidth between NE and datacentre or traffic node if the capacity is limited. Of course, depending on the load on the NE, there could be a solution to have one virtualised high performance server to accommodate all the required applications including NE as VNF. This would probably require additional HW to ensure availability.

The current way of selling and delivering the monitoring product to the customer is not very flexible, as the monitoring product is delivered with specific HW. This makes the delivery of the monitoring product longer with the HW and CSP can get the COTS HW cheaper as they usually buy the HW in bigger amounts for their datacentres and cloud environment to support different applications and purposes. Delivery of the specific HW might take from four to eight weeks, which can be avoided with the cloud. HW delivery leads to the situation that the monitoring product as bundled with hardware is much more expensive compared to a situation where the monitoring product would be sold only as software to the customer and the customer would buy the required hardware. With cloud, this hardware is not delivered and the CSP can buy the hardware that they want as stated in previous chapters that clouds can be based on COTS, which are cheaper than specific hardware. There is also possibility that CSP does not need to buy more HW for their cloud if there is enough capacity for the monitoring product or if they have spare HW to be just added to the cloud, and this would lead to very short delivery time.

Because of Microsoft licensing restrictions, acquiring the OS depends on the CSP and their contract with Microsoft. In this situation, the vendor should have the control over the OS so that no one in the CSP would log into the OS in the VM and make changes that could possibly disable the monitoring product. The target would be that VNF Management and Orchestration applications would take care of the VMs and OSs. One option, in case that the control over OS would need to be on the CSP because of licensing requirements, could be the creation of an installation server. This server would hosts scripts to configure or install the OS and the monitoring product. Then the CSP could initiate the installing of the monitoring product with a script that would fetch the monitoring product’s installation files from the vendor’s server, and then the licences to the OS and the monitoring product from the CSP’s server.

Going towards delivering only SW reduces vendor lock-in due to reduced upfront HW investments as it is easier to change the SW after licensing period. This creates the opportunity to capture new market share. On the other hand, this works also in the other way around and makes it even more important to keep the product ahead of competition.

Table 1 summarizes properties of the monitoring product when moving to the cloud. Plus signed properties helps when moving to the cloud and minus signed are the ones that create challenges towards the cloud maturity.

Table 1: Plus and minus properties of the monitoring product for the cloud.

+	Possibility to scale.
+	Possibility to install only needed and desired components.
+	Security.
+	Location distribution.
-	Tightly coupled components.
-	Stateful components.
-	Install and configure only through GUI.
-	Microsoft Windows® OS.
-	Some of the high performance capabilities tied to a specific HW.
-	SW and architecture maturity brings legacy. The product has been long in the market.
-	Use of database.

7.1 Implementation related to theory

Evaluating this cloudification on maturity model, the beginning or starting point would be having the product work in the cloud. This would mean that the product would not get all the benefits that the cloud is capable of giving. The starting point could mean these properties:

- Installation of the product is completely manual or partially automated.
- Configuration of the product is manual.
- Scaling would be manual, even though the NEs would be dynamically scaled.
- Management, maintenance and upgrading of the product are manual.

When the product matures enough and all of the benefits of cloud can be supported, it can be declared that the monitoring product is fully cloud compliant and the functionalities according to cloudification maturity model are fulfilled and the good practices from chapter 3.2 are implemented to the monitoring product. The properties would be fully automated life cycle with:

- Completely automated installation.
- Completely automated configuration, with the functionality that the product would understand its purpose and task with minimal information from the cloud management application and by communicating with the NE that it is monitoring.
- Scaling in and out would be automated, and scaling would be handled by a cloud management application.

- Management, maintenance and upgrades would be done completely automated after initiated either by the operator or by the cloud management application.

As stated previously, it is not feasible to jump straight to the nirvana on the cloud maturity model with a legacy application. This takes time and is costly due to rewriting of parts of the application. When the development of this product was started, there was not considered to have huge architectural decision on modularity. Many of the functionalities are tightly coded together and separating them will take time.

In this thesis the first step of the maturity model was achieved and the basic functionality in the cloud is confirmed. As stated previously, the capacity statement is almost impossible to do. However, as a part of separate product virtualization project a comprehensive dimensioning guideline was created. Contract or SLA between CSP and the vendor will have some kind of capacity statement to make sure that the monitoring product will work and benefit the CSP. But this will mean that the capacity statement is dependent on the resources given to the product. Certain level of functionality can be guaranteed if the given resources fulfill the requirements.

Now that the monitoring product is possible to run in the OpenStack cloud environment, the development to implement elasticity and automation will continue. The order of steps and decisions on how to proceed and with what time schedule will be decided by the architects and product management.

REFERENCES

[1] S. A. Ahson, M. Ilyas, Cloud Computing and Software Services: Theory and Techniques, Auerbach Publications, 2011, 456 p.

[2] Amazon EC2 Instances, Amazon, Available: http://aws.amazon.com/ec2/instance-types/

[3] Amazon EC2 Product Details, Amazon, Available: http://aws.amazon.com/ec2/details/

[4] N. Antapoulos, L. Gilliam, Cloud Computing: Principles, Systems and Applications, Springer, 2010, 385 p.

[5] AWS Import/Export, Amazon, Available: http://aws.amazon.com/importexport/

[6] R. Buyya, J. Broberg, A. M. Goscinski, Cloud computing: Principles and Paradigms, John Wiley & Sons, 2011, 664 p.

[7] Cloud-Init for Windows instances, Cloudbase Solutions, Available: http://www.cloudbase.it/cloud-init-for-windows-instances/

[8] Cloudification, Wiktionary, Available: http://en.wiktionary.org/wiki/cloudification

[9] Compute Nodes, OpenStack Manuals, OpenStack Operations Guide, OpenStack, Available: http://docs.openstack.org/openstack-ops/content/compute_nodes.html

[10] Conceptual architecture, OpenStack, Available: http://docs.openstack.org/admin-guide-cloud/content/conceptual-architecture.html

[11] Designing for the cloud, OpenStack, Available: http://docs.openstack.org/arch-design/content/designing-for-the-cloud.html

[12] ETSI NFV White Paper, ETSI NFV ISG, 2012, Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[13] ETSI NFV White Paper Update, ETSI NFV ISG, 2013, Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[14] ETSI NFV website, Available: http://www.etsi.org/technologies-clusters/technologies/nfv

[15] ETSI website, Available: http://www.etsi.org/about

[16] C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications, Springer, 2014, 393 p.

[17] Flavors, OpenStack, Available: http://docs.openstack.org/openstack-ops/content/flavors.html

[18] Getting Started, GigaSpaces Technologies, Available: http://getcloudify.org/guide/3.1/quickstart.html

[19] L. Grandinetti, O. Pisacane, M. Sheikhalishahi, Pervasive Cloud Computing Technologies: Future Outlooks and Interdisciplinary Perspectives, IGI Global, 2014, 325 p.

[20] Intel Corporation, End to End Cloud Computing, Intel Press, 2012, 200 p.

[21] How can I use an OpenStack cloud?, OpenStack, Available: http://docs.openstack.org/user-guide/enduser/intro-user.html

[22] HypervisorSupportMatrix, OpenStack, Available: https://wiki.openstack.org/wiki/HypervisorSupportMatrix

[23] K. Jamsa, Cloud Computing, Jones and Bartlett Learning, 2013, 342 p.

[24] M. Kavis, Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS), John Wiley & Sons, 2014, 224 p.

[25] KVM – Kernel Based Virtual Machine, Red Hat, Inc., 2015, Available: http://www.redhat.com/en/files/resources/en-rh-kvm-kernal-based-virtual-machine.pdf

[26] Manage IP addresses, OpenStack, Available: http://docs.openstack.org/user-guide-admin/content/manage_ip_addresses.html

[27] P. Mell, T. Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, 2011, Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[28] H. T. Mouftah, B. Kantarci, Communication Infrastructures for Cloud Computing, IGI Global, 2014, 583 p.

[29] Network Functions Virtualisation (NFV); Architectural Framework, ETSI NFV ISG, 2014,Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[30] D. Rountree, I. Castrillo, The Basics of Cloud Computing: Understanding the Fundamentals of Cloud Computing in theory and practice, Syngress Publishing, 2014, 174 p.

[31] A. Silberschatz, P. Baer Galvin, G. Gagne, Operating System Concepts Essential, Second Edition, John Wiley & Sons, 2014, 781 p.

[32] OpenStack Cloud Administrator Guide, OpenStack, 2015, Available: http://docs.openstack.org/admin-guide-cloud/admin-guide-cloud.pdf

[33] OpenStack: The Open Source Cloud Operating System, The OpenStack , Available: https://www.openstack.org/software/

[34] Over-Provisioning and Service Offering Limits, CloudStack, Available: https://cloudstack.apache.org/docs/en-US/Apache_CloudStack/4.2.0/html/Admin_Guide/over-provisioning-service-offering-limits.html

[35] Red Hat Enterprise Linux 6, Virtualization Host Configuration and Guest Installation Guide, Red Hat, Inc., Available: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/chap-Virtualization_Host_Configuration_and_Guest_Installation_Guide-Para_virtualized_drivers.html

[36] B. Sosinsky, Cloud Computing Bible, John Wiley & Sons, 2011, 528 p.

[37] A. Tseitlin, G. Orzell, Netflix Operations: Part I, Going Distributed, 2012, Available: http://techblog.netflix.com/2012/06/netflix-operations-part-i-going.html

[38] Understanding Answer Files, Microsoft TechNet, Available: https://technet.microsoft.com/en-us/library/cc749113%28v=ws.10%29.aspx

[39] J. Varia, Architecting for the Cloud: Best Practices, Amazon Web Services, 2011, Available: http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf

[40] T. Vesterinen, MSc (Tech), System Architect, Tampere. Interview on 28.04.2015

[41] What is Sysprep? Microsoft TechNet, Available: https://technet.microsoft.com/en-us/library/cc721940%28v=ws.10%29.aspx

[42] Windows Server Installation Options, Microsoft TecNet, Available: http://technet.microsoft.com/en-us/library/hh831786.aspx