Cloud Computing and Microservices

Fabian Wildgrube

Sommerakademie in Leysin, August 2016

Abstract

Cloud Computing, i.e. the the idea of accessing off-premise IT-infrastructure, platforms and ready-to-use applications via the internet (or another network) rather than owning and managing these on premise, is one of the fastest growing IT-trends of the last ten years. With its three levels of "as a Service" offers (Infrastructure, Platform, Software) developers as well as end-users benefit from the low cost, flexible and scalable environments Cloud Solutions create. Hand in hand with the Cloud go Microservices. This architectural approach breaks down a program into a number of separate tasks or responsibilities that are then implemented as independent services that communicate through a very simple protocol like HTTP resource API. This allows developers to work in small, independent teams speeding up development as well as enabling independent scaling of each service, which is more efficient than working with one big monolithic application that holds every aspect of a program within it.

Contents

1 Introduction

"The Cloud" – one of the buzz words of the last couple of years in the IT industry. Companies are moving into the Cloud, everybody has their personal Cloud, Cloud providers are popping up all over the web. However, it seems that no one really has a precise definition of what the ominous "Cloud" actually is. This paper will not try to give an all-embracing or comprehensive definition of the term "Cloud" but it will try to lay out and explain the principles and technologies that work together to create the quite heterogeneous set of services known as "The Cloud".

Another buzz word this paper will look at is "Microservices". This new architectural approach to programming large, fast growing applications has gained a lot of popularity since 2015 but even more so than with the Cloud it seems there is no clear definition of the term. Again this paper will try to explain the core principles of this newly rediscovered programming architecture and show why Microservices are so perfectly predestined to be used in conjunction with Cloud Solutions.

2 The Cloud

Before we take a closer look at the technologies involved we will give a short definition of Cloud Computing, which will be the basis of what this paper discusses:

> Cloud Computing describes the idea of accessing off-premise IT-infrastructure, platforms and ready-to-use applications via the internet (or another network) rather than owning and managing these on premise. *[1]*

2.1 The basic idea

The "Cloud" as we know it today was invented at Amazon in 2006. Amazon was growing fast and needed to scale their infrastructure to meet the demands. Especially around Christmas time the website reached peak usage like never before. However, after Christmas Amazon's new data centers weren't needed anymore – average traffic in the web shop was way below peak traffic. [2]

It was then that they developed the idea to share their infrastructure with other companies and charge them for it. Companies were able to take advantage of Amazons huge

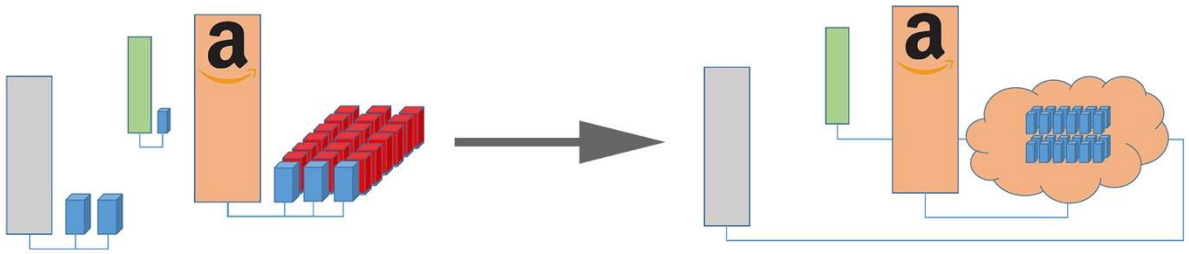

FIGURE 1: MOSTLY UNUSED INFRASTRUCTURE IS RENTED OUT TO OTHER COMPANIES

infrastructure without having to invest in it from the ground up but rather paying only for what they actually used - Cloud Computing was born. Since then Amazon Web Services has become one of the biggest Cloud-Service providers and has started one of the big IT-revolutions of the 21st century. And their idea is what still lies beneath all of the Cloud – using infrastructure (i.e. servers) hosted by another company somewhere in the world, and accessing one's data or software via the internet.

So using the Cloud basically means "outsourcing" all of the maintenance of servers, cooling, power supply, and so on, to another company who in turn charges for the use of their infrastructure. This means a company doesn't have to invest in expensive hardware to try out a new business tool anymore – they can just rent what they need for the time they need it. Which is another key feature of Cloud Computing: Flexible, demand-based adjustments of the service one is using. This goes hand in hand with the usually subscription-based or completely usage-based pricing of the offers. The pricing on these offers is usually quite moderate – allowing businesses to get started with new tools without taking huge financial risks. However, at a certain scale, Cloud solutions might actually become more expensive then managing the infrastructure and servers on premise, so one should always carefully calculate the cost of different solutions – especially considering long term developments.

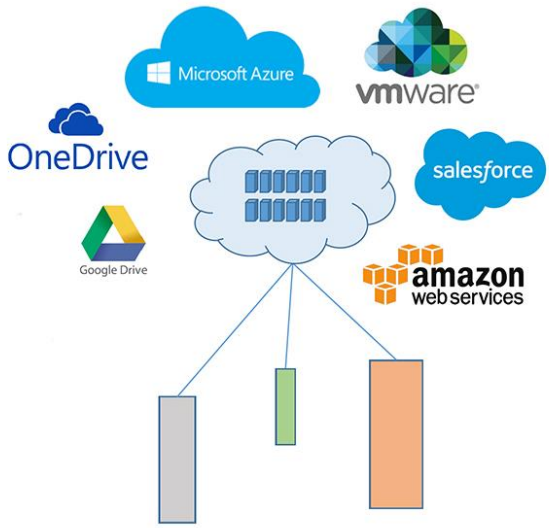

FIGURE 2: POPULAR CLOUD-SERVICES PROVIDERS

To sum up, "The Cloud" could be described as follows: [1]

- Several people or companies share the resources offered by a central provider
- Access via the internet
- Flexible, demand-based adjustments (Peak-Coverage) possible
- Subscription- or Usage- based pricing (flexible)

2.2 Different levels of Cloud Computing

Cloud Computing is usually divided into three layers, each adding another level of abstraction for the end-user. At the bottom of the "Cloud pyramid" is Infrastructure as a Service (IaaS). With these services the cloud provider offers the customer a running server instance[1] with physical maintenance (i.e. power supply, cooling and network access are provided) but no software maintenance. This means the customer has to take care of installing and updating an operating system and any software he needs himself. This is mainly for professionals who don't want to deal with physical maintenance but need to be in control of everything else on their machine. [1]

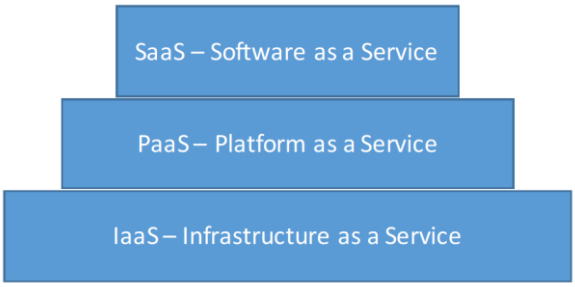

FIGURE 3: THE 3 LAYERS OF CLOUD COMPUTING

The next layer is called Platform as a Service (PaaS). Abstraction is added as the costumer does not need to take care of installing or updating the OS and is provided directly with a software platform (for example a JVM running on a server instance). Most of the time IDEs, a version-control system and collaborative tools are offered by the provider as well. This layer of the Cloud is mainly aimed at (web-application) developers. [1]

The most popular (and probably most widely known as "The Cloud") layer is Software as a Service (SaaS). Another level of abstraction is added and the costumer now only books instances of an application that he can access with his browser (or download to his local machine, but only use with a license provided by the Cloud host). Examples for this kind of service might be

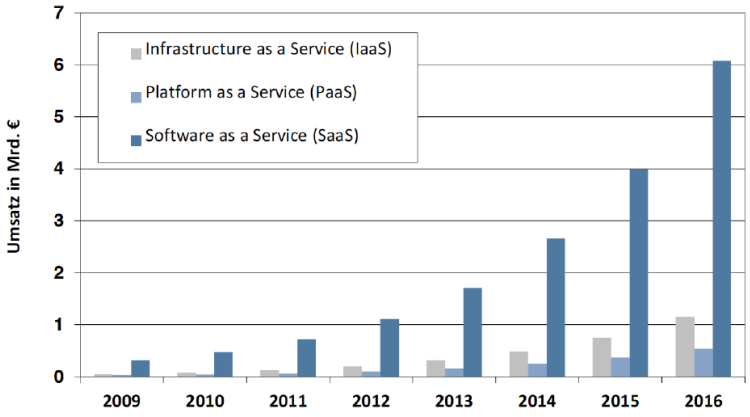

FIGURE 4: VOLUME OF SALES OF DIFFERENT CLOUD SERVICES **[4]**

[1] An instance might be an actual physical server, but is usually a virtual server. Since most of the time servers are only used at 12-18% capacity [4], multiple instances can easily be run on one physical machine.

Microsoft Exchange Mail-Accounts, Adobe Creative Cloud Software and many more. This layer of Cloud Computing is clearly aimed at the 'consumer', i.e. the non-IT-affiliated end-user. SaaS is also the most popular of the layers by far, as figure 4 shows. [1]

Another level of Cloud Services to consider is data storage which is often overlooked as it is usually included in the service levels mentioned above. However, the more data one wants to store in the Cloud the more expensive it becomes. Also, transferring a lot of data over the public internet can become a hazard in time-critical applications, which makes data a big factor of the decision about moving "into the Cloud".

2.3 Different Cloud Solutions

Another dimension to Cloud Services is how they are accessed. Depending on the level of data security one needs the 'typical' public Cloud might not fulfill certain standards, which is why there are several different alternatives.

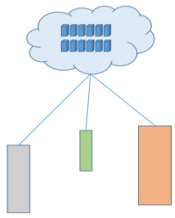

FIGURE 5: A PUBLIC CLOUD ARCHITECTURE

Public Cloud Services are publicly available through the internet. This means anyone can use them – of course the accounts are usually password protected, but the connection to the cloud provider might not be encrypted (or not very strongly). Also the cloud provider might store the data anywhere in the world, although this depends heavily on the provider[2]. [1]

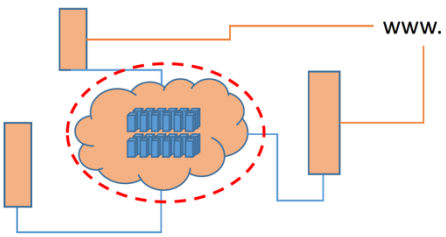

FIGURE 6: A PRIVATE CLOUD

Private Cloud Solutions are the exact opposite as the data center(s) they run on are owned and managed on premise by the company using them. For example, a company might have several different branch offices which are all connected to the central data center. This data center implements cloud-like services such as SaaS to enable the employees to work with the data in a comfortable way. The obvious advantages of a private cloud like this are that the company is in control of the location and security of the data. However, maintenance is not outsourced to a provider anymore and scaling these private cloud solutions won't be possible as fast since the company might have to invest in and set up new servers which takes time. [1]

[2] Amazon webservices for example offer to store one's data in certain regions of the world in so called 'availability zones', guaranteeing that the data is stored in data centers located in these regions.

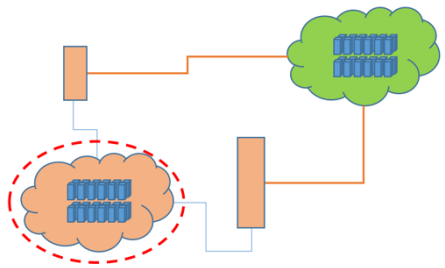

FIGURE 7: A HYBRID CLOUD SOLUTION

Another approach to bridge this conflict between control over sensible data and the advantages of outsourcing maintenance to a provider is a so called *Hybrid Cloud*. The costumer splits up his data and places sensible data and services in a private Cloud but uses public Cloud services for everything else. This approach tries to take advantage of the flexibility and low cost of public Cloud services and only resorting to more expensive and laborious private solutions where absolutely needed. [1]

A new trend is the `Virtual Private Cloud`. These services are offered by public Cloud providers who react to growing concerns about data safety. Access to these Cloud structures realized through VPNs, Encryption, private IP-addresses as well as assigning a Virtual Local Area Network (VLAN) for every costumer [3]

2.4 Cloud Data Centers

Data Centers – the engines of the internet. Without servers running around the clock all over the world there would be no www and no cloud as well. These data centers are huge, both in computer power as well as physical size. They are also located all over the world (although a trend can be seen, that new data centers try to take advantage of e.g. colder climates in the northern polar circle or nearby reusable energy sources, such as hydro plants). Amazon for example published this map showing how widespread their data centers are located.

FIGURE 8: AMAZON DATA CENTERS AND AVAILABILITY ZONES
Source: https://aws.amazon.com/de/about-°©-aws/global-°©-infrastructure/

2.5 Is it worth it? – Pros and Cons

Although the Cloud may seem like the ultimate solution for the future one has to look carefully at every aspect of it and especially consider negative effects going into the Cloud might have. The Cloud prices per instance might (at a certain scale) actually not be cheaper than maintaining on premise servers. Also security, privacy and liability in case of failure are important factors that have to be looked at.

The most important and potentially most dangerous aspect of Cloud Computing is the provider. The costumer depends completely on the provider – be it uptime of services, fast internet connection, physical security of the servers, data privacy, loss of data, etc. Especially in Germany data privacy must not be overlooked. German laws regarding consumer protection and privacy are very strict and won't allow for example, storing certain data sets abroad – which might be a problem if the Cloud provider only hosts his services in the US. Also contracts with the provider have to be closely examined, especially considering that the provider might go out of business. What happens with the data of a costumer should be clearly stated and regulated.

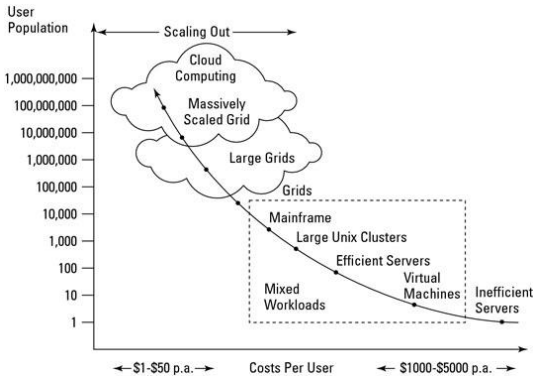

FIGURE 9: PRICES FOR SERVICES IN DIFFERENT IT-ENVIRONMENTS
http://media.wiley.com/Lux/03/274803.image0.jpg

However, the Cloud does offer huge improvements in cost efficiency and creates opportunities for businesses to expand fast without having to make big investments. Also it greatly frees up workflows, since collaborators on projects do not have to be at the same place anymore, as long as they can share their data through the Cloud. This helps enormously with productivity. The flexibility of scaling web applications or websites with a cloud service also helps improve efficiency – both monetarily as well as saving energy. Some might even argue energy efficiency is the most positive impact Cloud computing can have.

2.6 A closer look at the efficiency of Cloud Solutions

The problem with small data centers (for example two servers running in a back room at a small company) is their terrible power efficiency. An average server runs at about 12-18% of its capacity. [4] This means one could easily run multiple virtual instances on one physical machine – but in most data centers this isn't done. In huge Cloud data centers this is common practice, as well as efficient cooling systems and running new (more efficient) hardware. Cooling and upgrading old hardware continuously are issues that are not confronted as regularly in small data centers as they are in massive Cloud data centers.

Now this would not be a big problem if the Cloud made up for 90% of all data centers, but this is not the case. In the US only 4% of the electricity consumed by data centers are used by Hyper-Scale Cloud Computing. 49% percent (!) are eaten up by small- and medium-sized data centers discussed

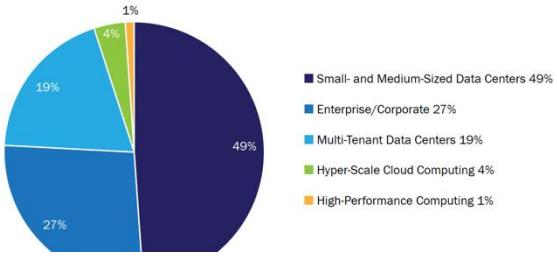

FIGURE 10: SERVER ELECTRICITY CONSUMPTION IN THE US **[4]**

above. This means that a huge part of the energy used for IT-Infrastructure simply goes to waste because of inefficient small data centers.

If one works with an average utilization of 15% in on premise small data centers it becomes clear that the Cloud is massively more efficient. An average cloud server runs at 65% of its capacity, which makes on premise roughly 50% less effective. On top of that Cloud data centers in most cases have more advanced and efficient cooling systems, making cloud solutions consume 84% less energy than on premise servers! So moving small businesses into the Cloud also has implications for the environment, which makes Cloud Computing a huge part of Green Computing of the future. [4] [5]

3 Microservices

Modularity, flexibility and independent scalability – these are keywords when describing the programming architecture called Microservices. Although as with the Cloud there is no precise, widely agreed-upon definition, a trend seems to develop to build web applications based on Microservices (Netflix, Amazon, …). To understand Microservices, one has to look at Monoliths first, though.

3.1 Prelude: Monolithic Architectures

Building a monolith means basically designing a program as one entity that handles everything – a closed box with different compartments inside, that can only be used (and scaled) as a whole. For example, a Java application is usually a monolith, where different classes handle the different aspects of the program but can only be used in the context of the complete program. [6]

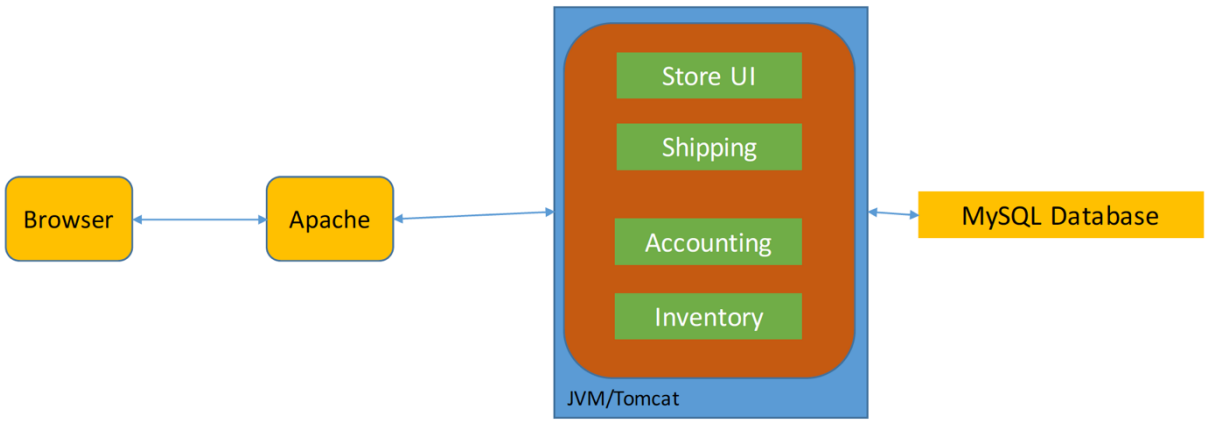

FIGURE 11: AN E-COMMERCE WEBSITE AS A MONOLITH **[6]**

This architecture is what we are used to and it does have several benefits. Firstly, deploying a monolithic application is rather easy, since there is only one file (e.g. a .war file), that contains the entire program. Secondly, most IDEs around today are tailored specifically to developing monoliths, which helps making development fast and comfortable. Debugging and testing tools are all in one place and chances are, that most developers know how to work with these tools. And lastly a monolith can be scaled, i.e. multiple instances of the program can be run behind a load balancer, to handle more traffic. However, this can also be a downside. If for example, only the StoreUI gets more traffic, but Shipping doesn't, scaling the monolith would mean running multiple instances of the whole program. So Shipping would be scaled up, although it wouldn't be necessary at all. On a large scale this can lead to massive loss of efficiency. [6]

There are even more downsides to using a monolithic architecture, though. The larger the program gets, the more code there is, which makes it more intimidating for new developers joining the team. Also the IDE becomes slower and deploying a massive monolith becomes painfully slow as well.

Modularity will be lost over time in a monolith since there are no hard boundaries between modules and developers will take lazy short cuts if they can. Another problem is that updating one part of the program (e.g. the UI) leads to redeploying the entire program, which in turn means downtime of the entire service for the time it takes to get the new version up and running. However, the two biggest problems with the monolithic architecture are that teams cannot work independently on different parts of the application and adapting new technologies becomes almost impossible. With a monolith one has to decide in the beginning, which programming language is used. Once this decision has been made usually there is no way of adding features in other languages that might be a better fit for specific problems.

3.2 The Microservices Architecture

Microservices try to address the issues that go with a monolith, in order to build flexible, maintainable, and easily scalable applications.

The basic idea of Microservices is to split up the application into a number of independent services, each running in their own process, having only one responsibility or task.

Communication between services is realized through a very simple protocol, for example HTTP resource API. The E-commerce Website from above could be realized as such: [7]

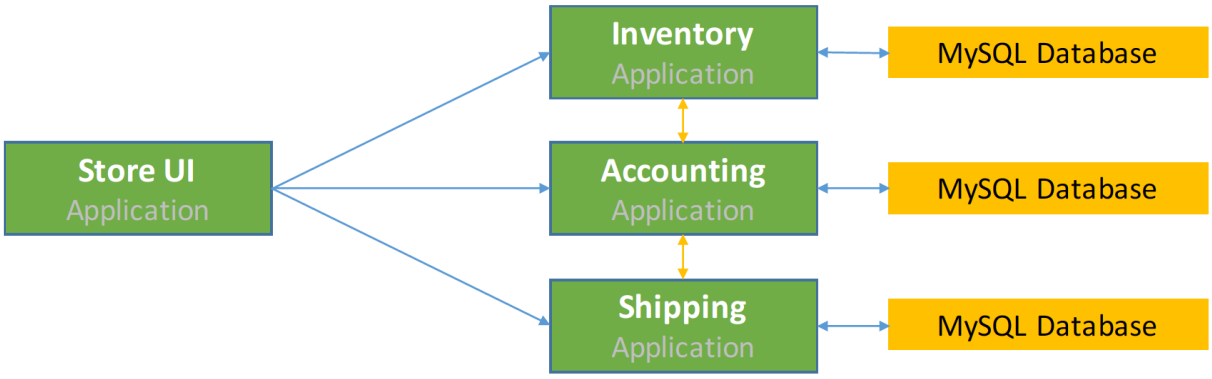

FIGURE 6: E-COMMERCE WEBSITE BUILT WITH MICROSERVICES **[7]**

Each task that makes up the store is in its own service and can only "talk" to the other services via their respective APIs.

To better define what Microservices really are, one can look at the key aspects of this architecture: [8]

- Independent services
- Dumb pipes – smart ends
- Single responsibility principle
- Each service has its own database

Most important is the independence of services. Each service has to be a runnable program, completely independent from other services. Although it might need to get information from other services it should never crash if one of the other services is not available. If this is achieved services can crash or be taken down for updating without the whole program braking down. What this also implicates is that redeploying and updating services can happen completely independently. [8]

"Dumb pipes – smart ends" describes the approach of keeping the communication protocol as basic as possible to only transmit data and do all the logic within the services. This helps to keep responsibilities very clear. Having a communication protocol with logic that interacts with the data would be counterproductive to the goal of splitting the program into its independent subsets. [8]

The single responsibility principle describes the fact that each service is only responsible for a very narrow set of functions, which are usually closely related. Basically each service does only one thing, but it does this thing very good. [9]

Lastly, to be truly independent each service has to have its own database. This adds complexity since consistency between databases has to be realized, however it is important. [8]

Using Microservices can be very beneficial for developing and maintaining a program, which is why huge companies like Amazon and Netflix have switched entirely to Microservices architectures in the last years. To understand why, one has to look at the benefits but also the downsides of Microservices

3.3 Pros and Cons of Microservices

There are downsides to Microservices and it doesn't always make sense to use them. Especially on a large scale one might end up with a couple hundred services which all have to be organized and tested. If there is no proper automatic system in place things can get really confusing and chaotic really fast. But automated tests (and deployment) are overhead, which has to be managed as well. [10]

Microservices also have to be designed for failure. This means that a service has to handle the failure of another service very well without breaking or crashing. If failure isn't handled gracefully a problem in one service could ripple through the entire program and cause extreme damage. Developers must really be aware of that and program their failure handling to prevent it. Hand in hand goes the issue of updating a service without breaking its so called "customer services"[3]. The API of a service should be backwards compatible until all customers have adapted the updated API. [10]

Perhaps the biggest challenge with a Microservices architecture is to find good modularity, i.e. to clearly identify the tasks and responsibilities within a program and divide them into separate services. If done wrong in the beginning it can become even messier than a monolith.

However, Microservices do have a lot to offer in return. Firstly, each service is rather small since it handles only one very specific task. This makes it easier for new developers to enter the team and start working on the service right away. Service failures won't affect the entire system, as long as each service handles failure of other services it interacts with well.

[3] A customer service is a service that accesses information from the „host service" via the host service's API.

Deploying new versions of a service is fast and there is no need to redeploy the whole program.

Because of the independence of services, teams can work separately at the same time, which greatly speeds up development. Also Modularity is preserved much more than in a monolith, since there are no "shortcuts" between services, everything has to be accessed via the services' API.

Very freeing is the possibility to use different platforms or programming languages in one program. Each service can be written in the language that best suits its requirements, which makes adapting new technologies a lot easier than with a monolith, where one commits to a specific platform at the beginning. [11]

Another huge benefit of Microservices is that they can be scaled independently. If, for instance the Store UI of the example above has an increase in traffic, Shipping might not because not everybody actually buys something in the store. Microservices can now be scaled accordingly, meaning only the Store UI service will run multiple instances to handle the traffic but all the other services will continue to run at their current state until traffic hits them. With big applications the possibility to scale each service individually and not the entire program can save huge amounts of computing power and disk space, etc. [8]

3.4 Microservices in Real Life

One of the first companies to adapt Microservices and eventually switch entirely was Netflix[4]. In 2008 their streaming service broke down completely due to a missing semicolon. [12] This was due to their monolithic architecture. In 2009 they started experimenting with Microservices by outsourcing movie encoding as a separate service and in 2010 continued to convert their monolith by moving Sign-Up, Movie selections, TV-selections and device configuration into separate services. By December 2011 all of Netflix ran on Microservices (500+). They are hosted by amazon web services – a Cloud provider.

3.5 Microservices and the Cloud

Because Microservices scale so well independently, running a Microservice program in the Cloud makes a lot of sense. The Cloud offers the option to scale up resources very fast as discussed in chapter two. Now a program such as Netflix will have differing traffic throughout one day – most people tend to watch TV in the evening. However running all services in many instances all the time would be very inefficient. So taking advantage of adaptive hardware scaling makes sense on its own. On top of that Netflix can scale only the

[4] Netflix is a video streaming service, which hosts (and produces) movies and series that can be viewed on demand. In July 2016 Netflix announced 83 million users worldwide.

services that actually get a whole lot of traffic independently (i.e. first Sign-up, then selection and then video player) which helps reduce cost and saves resources.

4 Conclusion

Microservices and the Cloud – two trends that have both transformed the IT world and continue to do so as they become more widely spread and implemented. Cloud Computing simplifies expanding a company's web-services or business tools while keeping the cost very low. As internet connections become faster and more and more devices are connected to the internet of things, Cloud Solutions will become even more popular and a possible development might even be, that most consumer products will become mere displays that push data to the Cloud where the operations and calculations are done and only the results are sent back to the devices. From an energy-conscious point of view this would probably be a positive development as huge Data Centers are more efficient and can be built close to reusable energy sources such as hydro power plants or windfarms (as Amazon and Facebook have already done). However, centralizing all computational power within a few hundred data centers would also create a monopoly that, looking at today's IT market, will probably lye in the hands of a few extremely powerful companies. As with all new technologies we have to carefully consider all the consequences and act in a way that is sustainable in the long term. The Cloud (in conjunction with Microservices) offers great improvements in performance, cost and energy efficiency there is no doubt it will continue to grow and be used in every aspect of our lives.

5 Bibliography

[1] T. G. Peter Mell, The NIST Definition of Cloud Computing, U. D. o. Commerce, Ed., National Institute of Standards and Technology, 2011.

[2] Amazon, "About AWS," 2016. [Online]. Available: https://aws.amazon.com/de/about-aws/. [Accessed 26 09 2016].

[3] M. Rouse, "searchDataCenter.de," TechTarget, 08 2013. [Online]. Available: http://www.searchdatacenter.de/definition/Virtuelle-Private-Cloud-Virtual-Private-Cloud-VPC. [Accessed 10 09 2016].

[4] P. D. Josh Whitney, "Data Center Efficiency Assessment - nrdc.org," 08 2014. [Online]. Available: https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf. [Accessed 26 09 2016].

[5] Amazon, "amazon.com - AWS & Sustainability," Amazon Web Services, 2016. [Online]. Available: https://aws.amazon.com/de/about-aws/sustainability/?nc1=f_cc. [Accessed 26 09 2016].

[6] C. Richardson, "microservices.io - Monolith," 2014. [Online]. Available: http://microservices.io/patterns/monolithic.html. [Accessed 26 09 2016].

[7] C. Richardson, "microservices.io - Microservices," 2014. [Online]. Available: http://microservices.io/patterns/microservices.html. [Accessed 26 09 2016].

[8] M. Fowler, "martinfowler.com - Microservices," 25 03 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html. [Accessed 26 09 2016].

[9] R. C. Martin, Agile Spftware Development, Principles, Patterns, and Practices, Pearson Education, 2003.

[10] S. Newman, *Principles of Microservices,* Devoxx, 2015.

[11] M. Fowler, "martinfowler.com - Microservice Trade-Offs," 01 07 2015. [Online]. Available: http://martinfowler.com/articles/microservice-trade-offs.html. [Accessed 26 09 2016].

[12] J. Piela, "ProgrammableWeb - Why Netflix moved to a Microservices Architecture," 02 04 2016. [Online]. Available: http://www.programmableweb.com/news/why-netflix-moved-to-microservices-architecture/elsewhere-web/2016/04/02. [Accessed 26 09 2016].