Last update: 04/05/20

Service-Oriented Architecture and Cloud systems

(Summary, notes, and extracts from relevant papers[1])

(This is a living document that will be updated every week until the end of the course)

Table of Content

[1] The document contains paragraph extracted from papers and students reports I propose as reference for the lectures. Due to lack of time, I did have the time to edit the text further.

1. Service-Oriented Architecture and Web services

We started the first lecture by pointing out a couple of important points

What can impact the design of software architecture: Technology, Application requirements, and best practices and patterns (state-of-the-art). Later, we argued that the "monolith" approach to design software architecture does not enable the evolution of the architecture, the maintainability of the software over time, … I gave the example of the Parallel Ocean Program. We introduced a simple model which is a kind of evolution of the client/server architecture. We broke the tight connection between the server and the client and introduced the concept of a "**broker**" which enable a client to find the appropriate services. The new architecture will only work if we (1) "standardize" the communication between the client and the server (2) hide the implementation of the service from the client. The latter is achieved by introducing an abstract interface which only tells the client how it can invoke the service. This approach helps to enforce the scoping of the services and enable separation of concern. Multiples services can be used; each service could be developed separately and easily integrated into the new architecture enabling the evolution of the architecture and its maintainability. We also mentioned very briefly the evolution of this simple model toward a finer grain model which enforces through best practices and patterns, even more, the independence of the services (share-as-little as possible), and the separation of concern. We finished the first lectures, by talking briefly about the Web Service implementation of the service-oriented architecture.

Something I did not touch upon, nor the IBM white paper (reference 1 in canvas) has covered the "**state**" of a service. Are these services stateful or stateless? The state is discussed in the SOA model and left out to the implementation. In principle service are stateless, for a good reason which is to make the design scalable; however, it limits the usability of service as there no standard way to keep the state.

The idea of splitting a program into apart and assign the role of a client or server to each part known as **Client-Server** does not lead to a very **sustainable and maintainable** software over the years. There is a need to reconsider the way we design distributed software architecture to enable reusability of existing code, maintainability of the code (changing part of the code without having to break the entire application), and integration (interoperability) new components/functions and which might have been designed and implemented in a different context.

Service-Oriented Architecture is a paradigm (see definition in the slides) which can be regarded as an evolution of the Client/Server architecture. This approach to design a distributed application/system introduced the concept of a new component with the simple role to help locate the appropriate services. There are many implementations of the Service-Oriented Architecture. However, by far the most popular one is the ***Web Services implementation*** because of the use of mature and well-established technologies like HTTP, SOAP, XML, …

- **Service Mesh**[2,3] while microservices are built independently, communicate with each other, and can individually fail without escalating into an application-wide outage. The

[2] https://cloud.google.com/architecture/service-meshes-in-microservices-architecture
[3] What's a service mesh? https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh

logic governing communication *can* be coded into each service without a service mesh layer—but as communication gets more complex, a service mesh becomes more valuable. A service mesh is that it takes the logic governing service-to-service communication out of individual services and abstracts it to a layer of infrastructure. **Service Mesh** is a way to control how different parts of an application share data with one another. A **service mesh is a dedicated infrastructure layer** built right into an app. This visible infrastructure layer can document how well (or not) different parts of an app interact, so it becomes easier to optimize communication and avoid downtime as an app grows

The web services come with several tools and frameworks that make the development of web services very easy. There is no need to know anything about SAOP, XML, HTTP … to be able to implement a Web service. A pre-requisite knowledge of these technologies is optional for designing Web services, **however**, a **minimal** knowledge will prevent you from designing web services that lead to low performance or do not scale with the size of the problem. Anyway, to develop web services, you just need to select the development environment and the language you like to program a web service (there are plenty of open-sources IDE which supports developing Web services).

Having said that, it worth to mention that Service Oriented Architecture and particularly web services implementation is not a magic stick; it also has its pitfalls:

- What you win in **portability** you lose in **performance**. Service-Oriented Architecture is not meant for real-time application
- By definition services are **stateless** and thus any application that requires stateful services will not be straightforward to design using services, the state in Web services has not been standardized and thus each system has its way to maintain the state of the application
- The technologies chosen for Web services like SOAP, XML **are not suited to move large data sets**. So, any service which handles directly in the SAOP message GB will fail.

- Web services come in three different flavours: **SOAP**[4] (historically the most common ones), **WSRF**[5,6] – services (implement stateful services on top of standard Web services), **REST**[7] services. As of today, SOAP services are becoming less and less popular. However, there are many legacy services which have been developed as a SOAP service and still do not have alternatives. By far, the most popular version of Web

[4] https://www.w3.org/TR/2007/REC-soap12-part1-20070427/
[5] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
[6] http://docs.oasis-open.org/wsrf/wsrf-primer-1.2-primer-cd-02.pdf
[7] https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

services is the REST which eventually on the long term will replace the SOAP implementation completely.

- What next? Web API which one to use?[8]

https://medium.com/metrosystemsro/enterprise-software-architectures-new-programming-paradigms-420deacd4da0

https://www.guru99.com/api-vs-web-service-difference.html#1

In my opinion, the main difference between the Service-oriented Architecture and the various implementation flavours is related to the manipulation of the **state** of the services.

The references (SOAP):

- Reference 1: homework (see discussion slides in canvas)
- Reference 2: is just an explanation of the 12 steps presented during the lecture (see slide 18-19)
- Reference 3: discusses in details the idea that SOA is not only about Runtime; it also encapsulates the design time. Also, in this paper in the section "SOA layers of Integration", you can read about the importance of the concept of integration the SOA approach, this is also was briefly during the previous lecture (see Slide 11).

The references (REST):

- Reference 1: homework (see discussion slides in canvas)
- Reference 2: A chapter in a book you can read for free in books.google.com. It covers the REST services in more details and could be regarded as extra material to the lectures on REST.
- Reference 3: Is a paper that focuses on the migration of the legacy application from SOAP to REST. This is something interesting to read about if migration is not possible then there is a lot to do to replace all SOAP legacy services that are deployed all over the internet

2. Distributed Systems: Clusters, Grids, and Clouds

Using resources beyond one datacentre is facilitated by Grid and Cloud approaches[9].

You have learned in your courses on Parallel programming how to program multicore architectures and multimode systems within a single cluster. What if the cluster does not have

[8] https://www.youtube.com/watch?v=NFw0HznpLlM

[9] Foster et al. "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop, 2008. GCE '08 , vol., no., pp.1,10, 12-16 Nov. 2008 doi: 10.1109/GCE.2008.4738445

enough resources computing and storage), you need to use resources beyond the cluster. In this case, you are getting in a different environment where resources are **no not under a single administrative domain** (**no central control**), the network is slower and not reliable, and the hardware is heterogeneous (different architectures). Developing software applications in such an environment is rather challenging. The Application developers will be spending more time on the system related issues than on the application itself. Among the issues we have mentioned today during the lecture:

- **Authentications/Authorisation**: how the application developers identify him/herself to the various systems.
- **Resources management**: how to execute his/her code on independent systems
- **Data management**: how to get the data in an efficient way at the right place at the right moment to be processed.

2.1 Grid systems

Developing applications in such an environment is **not impossible**, but it is rather challenging for most of the application developers. The Idea of the Grid and later the Cloud is to put software layer on top on the existing software stake of a cluster to make them behave like one giant cluster. The first attempt was to introduce a middleware known as the Grid[10,11] (1995-2006) which adopted the services approach to implement the required functions to realise this vision. In the lecture, we identify the 4 functions of the Grid Service Architecture namely: (**1**) ***Grid-Security infrastructure*** (which solves the authentication, authorisation issues) , (**2**) ***Grid-Information service*** (which acts as the broker of this service oriented architecture), (**3**) ***Grid-resources management service*** (which acts as meta-scheduler submitting the jobs to be executed to various clusters and composing the Grid), and (**4**) finally the Grid ***Data management services*** (which mainly abstract the physical location of data from the program and does an automatic mapping between the logical names used in the real files distributed over the Grid storage.

In Grid systems, you **do not pay for the resources** but you need **certificate** (X509 certificate[12]) signed by a Certification Authority[13] to use the distributed Grid resources (CPU and storage). Access to Grid resource is fixed by the virtual organization[14] you are or will be a member of. The Grid middleware abstracts the administrative and technical complexity of the distributed infrastructure from the end-users. However, the application has to be submitted as a job to the local queue on specific resources. Because of the heterogeneity of the Grid systems, likely a significant portion of the resources is not usable by a given precompiled application. The overhead of using Grid resources is quite high and thus you should only be used when you have a significant computation (a running job for a few seconds/minutes on Grid systems will not help you to achieve big performance)

[10] Ian Foster, Carl Kesselman, Steven Tuecke, The Anatomy of the Grid Enabling Scalable Virtual Organizations, https://arxiv.org/pdf/cs/0103025.pdf
[11] The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration https://www.globus.org/sites/default/files/ogsa.pdf
[12] What is X.509 certificate https://www.sslauthority.com/x509-what-you-should-know/
[13] DutchGrid: the Dutch Grid certification authority is https://ca.dutchGrid.nl/
[14] https://en.wikipedia.org/wiki/Virtual_organization_(Grid_computing)

2.2 Cloud systems

Cloud systems abstract completely the infrastructure (computing resources) from the application by introducing a virtualisation layer. Applications are executed in Virtual machines, which contains all the necessary packages needed to execute the application. Cloud systems can thus execute the virtual machines on almost all available computing resources. The access to Cloud resources often cost money (on public Cloud). Luckily you **only need to pay for what you use**. Because the resource provisioning model of Cloud systems is **very elastic,** you can request more resources when you need them and stop them when they are not needed any more (**calling up** and **down**)

At the beginning of the Cloud era, the Cloud systems worked **exclusively with VM**s, now public Cloud providers [15] introduce more and more containers to achieve process isolation. In general, Cloud middleware provides the following functionalities for the Cloud middleware:

- Providing access to a various Virtual image that can be customized by the application developers (a kind of VM repository or registry)
- Instantiating the VM and make them accessible to the application developers
- Sharing computing and storage resources among VM
- Managing the communication from and to the VM
- Enable horizontal scaling, the application developer can request VM to be instantiated on demand.

2.3 Cloud platform

Last time we have discussed in some details the Grid middleware and introduce very briefly the Cloud as an approach that offers the infrastructure, runtime and software as services. I hinted a bit at the difference between Grid and Cloud in terms of **abstraction** and **virtualisation**. I did present the main component of the Grid middleware, but we did not say anything about the Cloud middleware. As a distributed system Cloud should also have a middleware which coordinate the various components of the Cloud platform but what are the main functions of this middleware? Are these functions similar to the one for the Grid middleware? (resource management, Information management, security, and data.

Sempolinski[16] compared a couple of platforms which emerged as opensource platform and summarizes the entire Cloud computing stack starting for the hardware and OS up to the application level. Generic open-source Cloud computing systems rely on 5 fundamental elements:

- **The physical hardware**: the processors of the physical nodes to be used a computing node in Cloud systems need to have hardware extensions to run pure virtualisation,; otherwise, the overall system will be limited to para-virtualization only. We might get away without such extension and a Cloud system can be setup, but it greatly limits both

[15] https://www.zdnet.com/article/top-Cloud-providers-2019-aws-microsoft-azure-google-Cloud-ibm-makes-hybrid-move-salesforce-dominates-saas/

[16] Milojicic D., Liorente I.M., and Montero R.S., OpenNebula: A Cloud Management Tool, IEEE InternetComputing, Volume:15 Issue:2

the speed of the VMs and the flexibility available in choosing software components for the system

- **The network**. includes the DNS, DHCP and the subnet organization of the physical machines. It also includes virtual bridging of the network that is required to give each VM a unique virtual MAC address. This bridging is accomplished using programs such as bridge-utils, iptables or ebtables. Moreover, in addition handling the physical nodes, DHCP and DNS processes must be configured, in concert with the Cloud framework, to handle the MAC and IP addresses of virtual nodes as well
- **The third component is the virtual machine hypervisor**, (also known as a Virtual Machine Monitor or VMM). Popular opensource VMMs are Xen and KVM, and there are also a couple of commercial solutions as well like Vmware, and VirtualBox (also has an open-source version). **Hypervisors** provide a framework which allows VMs to run. In addition to the actual VMM itself, each of these Cloud frameworks relies on a library called libvirt, which is designed to provide a facility for controlling the start and stop of VMs. However, the libvirt abstraction is not perfect, as each VMM has unique options that must be set. As such, the inputs to libvirt can differ slightly depending on the VMM (and VMM version) used. For this and other reasons, the different Cloud frameworks support different subsets of the hypervisors.
- **The fourth component is an archive of VM disk images**. In order to run VMs, a virtual hard drive must be available. In cases where one is simply creating a single VM on a single physical machine, a blank disk image is created and the VM installs an operating system and other software. However, in a Cloud framework, where it is expected that hundreds of VMs will be started and shut down in a short amount of time, it is impractical to do a full OS install on each one. For this reason, each Cloud system has a repository of disk images that can be copied and used as the basis for new virtual disks. In any given Cloud, we must make a distinction between **template disk images** and **runtime disk images**. The template disk images are those stored in a disk image repository to be used for multiple VMs. When a VM is spawned, one of those templates copied and is packaged into a disk image appropriate for the given hypervisor. Usually, this involves adding a swap partition and padding the disk image to the appropriate size. It is the actual runtime image that is used by the virtual machine
- **The fifth component is the front-end for users**. There must be some interface for users to request virtual machines, specify their parameters, and obtain needed certificates and credentials in order to log into the created VMs. Some front-ends perform various types of scheduling by giving users an allotment of resources which they are not allowed to exceed. Other front-ends implement standard API such as EC2. We note that the front-end is one of the most customizable pieces of the entire system

The design and development of Cloud computing solutions bring several technical challenges to Cloud developers and operator. Two roles should be considered when designing Cloud systems: The User (who is concerned by the ease of use) and the provider (who is concerned by the toease of operations). These challenges can be grouped in three main areas: negotiation, decision, and operation.

- **The negotiation area, these are the challenges relative to how application developers interact** with the Cloud as well as the description of the Cloud offerings (API). It also includes the definition of the programmability level that the Cloud solution will offer
- **the decision area** copes with the main problem that Clouds faces behind the scenes: How virtual resources can be scheduled to meet user requirements? This mapping can is NP-hard and subject to on-going research. Rarely a user requests a single VM, it is often a complex request involving setting a Virtual network with a specific topology of virtual nodes (VM) and virtual links. Depending on the nature of the contract, a scheduling algorithm will cope with quality restrictions, **jurisdiction restrictions**, elastic growth, and so on. SLAs which are Not part of the Grid and difficult to implement because the rid was based on a kind of best-effort resource management, the only way to guaranty the availability of the resources was through advanced reservation which is not supported by all sites. But in the Cloud approach, SLA is something native because the whole model is based on payment.
- **The operation area** is associated with the enforcement of decisions and the communication between Cloud elements. The enforcement here covers the communication protocols and the configuration of Cloud elements. A communication protocol can be used to monitor and reserve resources in the Cloud. Cloud systems are composed of different elements like processing servers, switches, routers, links and storage components. Due to such heterogeneity, the communication between the decision-maker and certain components of the Cloud systems puts a challenge on Cloud design. Overall, existing Cloud solutions use Web Services to provide communication with processing and storage nodes, but many communication elements do not support such implementations. Thus, Cloud architects are using traditional traffic engineering[17] protocols to provide reservation of network elements. One possible idea to cope with this challenge is to use **smart communication nodes with an open interface to create new services in the node the emerging Openflow-enabled switches[18]**. Node communication is just one part of the problem; the other one is to configure this. Here, the recent advances in server virtualisation have provided several solutions for operators to benefit from

Between 2009-2013, there were a number of attempts to propose a taxonomy for Cloud systems. Rimal et all[19] propose a taxonomy to express the perspective of the users, it highlights a number of features that could be considered wen the users want to select a Cloud provider for their applications. David Hilley [20], proposes a taxonomy which aim at clarifying the relationships of the various Cloud computing products and which allow both consumers and service providers to assess their current and planned future offerings in light of their desired properties and marketplace positioning. David Hilley scoped as limited to lower-level offerings. In particular,

[17] Traffic engineering involves adapting the routing of traffic to the network conditions, with the joint goals of good user performance and efficient use of network resources
[18] [McKeown et al 2008]
[19] Rimal, B.P.; Eunmi Choi; Lumb, I., "A Taxonomy and Survey of Cloud Computing Systems," INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, vol., no., pp.44,51, 25-27 Aug. 2009 doi: 10.1109/NCM.2009.218
[20] David Hilley, Cloud Computing: "A Taxonomy of Platform and Infrastructure-level Offerings", College of Computing Georgia Institute of Technology http://www.cercs.gatech.edu/tech-reports/tr2009/git-cercs-09-13.pdf

"infrastructure" or "platform"- level services as well as relevant supporting services – this will include offerings like Amazon's EC2 or Google's App Engine. This taxonomy does not cover application-level offerings such as GMail, Google Docs, or Microsoft Online Services. Cordeiro et all. [21] …<TODO>. Sempolinski, P.; Thain[22] <TODO>

3. Virtualisation[23]

There is a clear distinction between the many types of virtualisation[24],[25] ranging from simple OS virtualisation to Full-virtualisation:

- (1) **Operating system level virtualisation** - uses isolation mechanisms to provide users with virtual environments similar to a dedicated server.
- (2) **Para-virtualisation** - requires the **guest operating system to be modified** to use a set of hooks to improve machine execution simulation – Para-virtualisation enabled by Type1 Hypervisor
- (3) **Full virtualisation** fully simulates all the hardware of the underlying device by providing a virtual machine)

3.1 Operating system level virtualisation (Nonvirtualizable vs virtualizable instruction)

There are many reasons for applications to access hardware: Receiving mouse and keyboard events, accessing devices, such as a FireWire DV camcorder, and driving a device from an application are just a few. Although only code that resides in the kernel can access hardware directly, the operating systems provide many services that allow applications to communicate with hardware from plug-ins, applications, shared libraries, and other code running outside the kernel. Most processors have different privilege levels[26]; that lets the OS interrupt, confine and kill processes without letting any user-level process affect any other.

There are instructions to handle the privilege level, some low-level hardware managing. Usually, these instructions are restricted to work only at the most privileged level; therefore, the OS has to run at this level, if you try to run low-level OS code in a lower privilege, those low-level instructions will fail. Ideally, the failure should trigger some 'real lower level' OS, that could make whatever magic is needed to create the illusion that the failed code is running on a bare metal machine. Unfortunately, on x86 **a few (17) instructions** either fail silently or trigger global failure, so they're referred as 'non virtualizable'[27].

The hypervisor managers VMM (virtual machine monitors) that host virtual machine VVM, or **hypervisor[28]:** The terms both go back to the '60s (when CS terminology was less

---

[21] Cordeiro, T.; Damalio, D.; Pereira, N.; Endo, P.; Palhares, A.; Gonçalves, G.; Sadok, D.; Kelner, J.; Melander, B.; Souza, V.; Mångs, J.-E., "Open Source Cloud Computing Platforms," Grid and Cooperative Computing (GCC), 2010 9th International Conference on , vol., no., pp.366,371, 1-5 Nov. 2010 doi: 10.1109/GCC.2010.77

[22] Sempolinski, P.; Thain, D., "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus," Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on , vol., no., pp.417,426, Nov. 30 2010-Dec. 3 2010 doi: 10.1109/CloudCom.2010.42

[23] What is virtualization: https://www.youtube.com/watch?v=g8UeGe-Fm1M

[24] Full and Para Virtualization http://www.unf.edu/~sahuja/Cloudcourse/Fullandparavirtualization.pdf

[25] https://www.virtuatopia.com/index.php?title=Main_Page

[26] https://en.wikipedia.org/wiki/CPU_modes

[27] Cpu virtualization: https://www.youtube.com/watch?v=ReUbbY13dvw

[28] https://www.youtube.com/watch?v=Q0XyphhfJXE

standardised than today), and traditionally refer to the same thing: the lowest (and most privileged) software layer implementing virtual machines. "Hypervisor" was IBM's term, others used virtual-machine monitor (VMM). Particularly on the traditional x86 architecture full virtualisation poses particular difficulties and has significant costs in complexity and performance. This minimum level of service required from each guest operating system must be coupled with resources available to dedicate to I/O operations, interrupt handling and input from other components which form a widely varying set of demands. Aside from hiding the hardware from a guest operating system the hypervisor's role are many. A hypervisor must protect **multiple guest operating systems from interfering** with each other: (1) preventing each from writing into each other's memory or disk space and **manage the sharing of resources** between many different operating systems each with different requirements running different applications. There are two types of hypervisors.

- Typer-1 (**bare metal/native hypervisor**) runs directly on the host machine's hardware[29]. **type-1 para-virtualisation,** there many hypervisors that are type-1, Wmware: ESXi, Xen, Microsoft Hyper-V
- Type 2 Hypervisor (**Hosted hypervisor)**[30] is a virtualisation layer that is installed above a host operating system (OS), such as Windows Server, Linux, or a custom OS installation. The host operating system has direct access to the server's hardware and is responsible for managing basic OS services. The Type 2 Hypervisor creates virtual machine environments and coordinates calls for CPU, memory, disk, network, and other resources through the host OS.

The distinction between the two types of hypervisors is important and a hosted hypervisor has a much higher inherent virtualisation cost due to the need for the VMM to go through many more layers of software[31]. The performance of native, bare-metal hypervisors, (since they run on the 'bare-metal' of the hardware) is generally much better than that of hosted hypervisors.

3.2 Full-virtualisation[32]

In a fully virtualized system, the guest OS is not modified and so the VMM must provide the virtual machine with a simulated (Virtual) device drivers, which are called by the guest OS to run "*the equivalent system calls*" on the host hardware, simulating hardware interrupts which the guest OS would normally receive from the hardware. In a native environment hierarchical, protection domain, called protection rings[33], are mechanisms to protect data and functionality from faults (fault tolerance) and malicious behaviour (computer security). This is generally hardware-enforced by some CPU architectures that provide different CPU modes at the hardware or microcode level. The virtual drivers of the VMM **translates kernel code** to replace nonvirtualizable instructions (which only run in ring 0) with new sequences of instructions that have the intended effect on the virtual hardware[34]. Meanwhile, user level code is directly executed on the processor for high performance virtualisation. Each VM monitor provides each

[29] install from UCB stick to mother board see http://www.youtube.com/watch?v=zCvAVUc6-Us
[30] https://www.youtube.com/watch?v=T3FWGd_jUoc
[31] https://www.youtube.com/watch?v=G5cQtGpAfqQ
[32] Full virtualization https://www.youtube.com/watch?v=CLR0pq9dy4g
[33] https://en.wikipedia.org/wiki/Protection_ring#Privilege_level
[34] https://www.youtube.com/watch?v=ppkzbhE83pY

VM with all the services of the physical system, including a virtual BIOS, virtual devices and virtualized memory management

3.3 Para-virtualisation[35,36,37]

In para-virtualisation the guest OS "know" that it is virtualized to take advantage of the functions (guest OS is modified). Guest operating systems require extensions to make API calls to the hypervisor. Paravirtualization is a new term for an old idea. IBM's VM operating system has offered such a facility since 1972 (and earlier as CP-67). In the VM world, this is designated a "DIAGNOSE code", because it uses an instruction code normally used only by hardware maintenance software and thus undefined. "In 2005, VMware proposed a paravirtualization interface, the Virtual Machine Interface (VMI), as a communication mechanism between the guest operating system and the hypervisor. This interface enabled transparent paravirtualization in which a single binary version of the operating system can run either on native hardware or on a hypervisor in para-virtualized mode. As AMD and Intel CPUs added support for more efficient hardware-assisted virtualization, the standard became obsoleted and VMI support was removed from Linux kernel in 2.6.37[38] and from VMware products in 2011"[39]

3.4 Hardware supported virtualisation

As discussed above the challenges in designing and implementing hypervisors are considerable. Software techniques have coped well with these problems and navigated them well. Another type of virtualisation emerged with the increasing popularity of hardware support for virtualisation has driven demand for the development of embedded microkernels and hypervisors. Hardware vendors developed new processor extensions to the x86 architecture, which helped to simplify virtualisation techniques. In 2005, both **Intel** and **AMD** added an extra mode to traps those **nonvirtualizable instructions**[40]. First generation enhancements include Intel Virtualisation Technology (VT-x) and AMD's AMD-V which both target **privileged instructions** with a **new CPU execution mode** feature that allows the **VMM** to run **in a new root mode below ring 0**.

- **Hardware hypervisors** mean the possibility of virtualisation in embedded devices and so we can run high level operating systems in devices embedded with virtual machine managers. Hardware hypervisors are more complex than their software counterparts, Virtualisation affords embedded systems with improved robustness, control over task scheduling and system-wide real-time event response as well as effective energy management, impacting heavily on how users interact with operating systems and virtual environments by running counter to current models of embedded systems design.
- **Microkernels** provide strategies like scheduling, device drivers and memory management and since these strategies are all implemented as individual user mode processes they can be treated as isolated components. Microkernels and their component-based architecture have become important concepts in running distributed services in a virtual environment.

---

[35] Para-visualization (part1) https://www.youtube.com/watch?v=1PYNcmZTjiE
[36] Para virtualisation (part2) https://www.youtube.com/watch?v=uoTPmCUfedU
[37] Para-virtualization(part3) https://www.youtube.com/watch?v=_Tltue_pa-o
[38] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=d0153ca35d344d9b640dc305031b0703ba3f30f0
[39] https://en.wikipedia.org/wiki/Paravirtualization
[40] https://www.youtube.com/watch?v=IbtXmvuFw6w

3.5 Memory virtualisation[41]

Beyond CPU virtualisation, the next critical component to virtualise is the memory. This involves sharing the physical system memory and dynamically allocating it to virtual machines. VM memory virtualisation is similar to the virtual memory support provided by modern operating systems. Applications see a contiguous address space that is not necessarily tied to the underlying physical memory in the system. The operating system keeps mappings of virtual page numbers to physical page numbers stored in page tables. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance.

To run **multiple VMs** on a single system, another level of memory virtualisation is required. In other words, one has to virtualize the MMU to support the guest OS. The guest OS continues to control the mapping of virtual addresses to the **guest memory physical addresses**, but the guest OS cannot have direct access to the actual machine memory. The VMM is responsible for mapping guest physical memory to the actual machine memory, and it uses shadow page tables to accelerate the mappings. The VMM uses TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access. When the guest OS changes the virtual memory to physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup. MMU virtualisation creates some overhead for all virtualisation approaches, but this is the area where second generation hardware assisted virtualisation will offer efficiency gains.

3.6 Device and I/O Virtualisation

The hypervisor virtualizes the physical hardware and presents each VM with a standardized set of virtual devices like the network card. These virtual devices effectively emulate well-known hardware and translate the VM requests to the system hardware. I/O Virtualisation involves managing the routing of I/O requests between virtual devices and the shared physical hardware. This standardization on consistent device drivers also helps with VM standardization and portability across platforms as all virtual machines are configured to run on the same virtual hardware regardless of the actual physical hardware in the system.

4. Containerization

Containers have (and will continue to) created a shift in the way we do computing from operations to scientific programming. This will influence the way you develop and deploy code in the future. "Often thought of as cheap VMs, containers are just **isolated groups of processes** running on a single host. That isolation leverages several underlying technologies built into the Linux kernel: **namespaces** (limit what a process can see like pid, net, mnt, ipc, …), **cgroups** (limit how man resources the process can use like CPU, memory, I/O, networks …), **chroots** and lots of terms you've probably heard before …"[42]. A container helps to package the application

[41] Memory virtualization full https://www.youtube.com/watch?v=UgVnzAGBeWI
[42] Source: Eric Chiang "Containers from Scratch" https://ericchiang.github.io/post/containers-from-scratch/

software and its dependencies and enables it to run on different computing environment[43]. The benefit of containers provides a viable choice for general application isolation, infrastructure deployment and packaging of software services, especially in I/O intensive applications. Additionally, the process, filesystem, and resource isolation layer bring the advantage of sealing shut an application along with all the global resources and environment dependencies. This prevents users from running into incompatibility issues that would otherwise render the entire application/job unusable until the required internal dependencies are addressed. Container-based virtualization has gained an increasing popularity in the scientific field as well, one of the most desired applicability of contains would be the **reproducibility** and **portability** aspect

Container-based virtualization, a "lightweight" version of the hypervisor-based virtualization, aims at mitigating the performance overhead and introduces a new set of features that prevail those of hypervisor-based virtualization technologies. Due to their ability to share resources with the host machine, containers are able to avoid some of the penalties incurred on the hardware level isolation and reach near-native performance when tested against CPU-intensive applications[44],[45]. Containers come with the advantage of achieving lower start-up times[46] than that of a traditional hypervisor-based virtualization. Containers handles processes, **filesystems**, **namespace** and **spatial isolation** provision hardware resources independently for each running instance.

In recent years, the focus has shifted towards the usage of container-based virtualization in HPC applications. These applications have a large number of processor-intensive tasks, thus requiring the usage of parallel infrastructures so that each job can be divided in multiple mini-batches and distributed across a network of connected machines. Once this workload would be encountered by a hypervisor-based virtualization technology, multiple performance penalties would emerge once executed on top of the hypervisor context needed for Hypervisor-based virtualization[47]. Containers have gained increasing attention in HPC applications due to the benefit of removing the hypervisor dependency, just-in-time compilation, the performance degradation and the slow booting times of VMs.

Container technologies such as Singularity[48], Docker[49], OpenVZ[50] and Linux containers (LXC) have rapidly contributed to the development and wide-spread of container-based virtualization mechanisms. Each of the technologies mentioned above implements their method of achieving process hardware and network isolation, while some focus on specific applicability in the

[43] https://www.docker.com/resources/what-container
[44] M G Xavier, et al. Perfor- mance Evaluation of Container-based Virtualization for High Performance Computing Environments. Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, (LXC):233–240, 2013. ISSN 1066-6192. doi: Doi10.1109/Pdp.2013.41.
[45] Stephen Soltesz, et al. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, pages 275–287. ISSN 0163-5980. doi: http://doi.acm.org/10.1145/1272998.1273025.
[46] Rajdeep Dua, A Reddy Raja, and Dharmesh Kakadia. Virtualization vs containerization to support paas. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 610–614. IEEE, 2014.
[47] M.G.Xavier, et al. .Performance evaluation of container-based virtualization for high performance computing environments. In 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pages 233–240, Feb 2013. doi: 10.1109/PDP.2013.41.
[48] https://www.sylabs.io/docs/
[49] https://www.docker.com
[50] https://openvz.org

industry, such as Docker[51], others focus on the portability containers across HPC environments, such as Singularity. A few points on Dockers, Singularity, and Kubernetes:

4.1 Docker

- requires a daemon process (docker engine) to run as root, could lead to security issues
- uses liblxc which later on was runC
- **isolation namespaces**: processes running within a container cannot see, and even less affect, processes running in another container, or in the host system
- **isolation network**: containers run on top of a bridge interface. Each container also gets its own network stack, meaning that a container doesn't get privileged access to the sockets or interfaces of another container
- **isolation Control groups**: implement resource accounting and limiting. They provide many useful metrics, but they also help ensure that each container gets its fair share of memory, CPU, disk I/O; and, more importantly, that a single container cannot bring the system down by exhausting one of those resources
- Uses copy-on-write (CoW) technology with both images and containers. CoW strategy optimizes both image disk space usage and the performance of container start times

4.2 Singularity:

- No dependency of a daemon can be run as a simple user
- Singularity achieves mobility by utilizing a distributable image format that contains the entire container and stack into a single file.
- User inside a Singularity container is the same user as outside the container.
- Singularity minimizes the number of virtualized namespaces. The goal is not full isolation but mobility, freedom and performance

5. Containers orchestration[52]

"Container orchestration **automates** the **deployment**, **management**, **scaling**, and **networking** of containers. Enterprises that need to deploy and manage hundreds or thousands of containers can benefit from container orchestration. Containers orchestration can help you to deploy the same application across different environments without needing to redesign it. And microservices in containers make it easier to orchestrate services, including storage, networking, and security.

Containers give your microservice-based apps an ideal application deployment unit and self-contained execution environment. They make it possible to run multiple parts of an app independently in microservices, on the same hardware, with much greater control over individual pieces and life cycles.

[51] https://linuxcontainers.org
[52] This section is extracted from the https://www.redhat.com/en/topics/containers/what-is-container-orchestration

Managing the lifecycle of containers with orchestration also supports DevOps teams who integrate it into CI/CD workflows. Along with application programming interfaces (APIs) and DevOps teams, containerized microservices are the foundation for Cloud-native applications."

Container orchestration tools provide a framework for managing containers and microservices architecture at scale. Many container orchestration tools that can be used for container lifecycle management. Some popular options are **Kubernetes**, **Docker Swarm**, and **Apache Mesos**.

5.1 Kubernetes

Kubernetes is an open source container orchestration tool that was originally developed and designed by engineers at Google. Google donated the Kubernetes project to the newly formed Cloud-Native Computing Foundation in 2015.

Kubernetes orchestration allows you to build application services that span multiple containers, schedule containers across a cluster, scale those containers, and manage their health over time.

Kubernetes eliminates many of the manual processes involved in deploying and scaling containerized applications. You can cluster together groups of hosts, either physical or virtual machines, running Linux containers, and Kubernetes gives you the platform to easily and efficiently manage those clusters.

More broadly, it helps you fully implement and rely on a container-based infrastructure in production environments.

These clusters can span hosts across public, private, or hybrid Clouds. For this reason, Kubernetes is an ideal platform for hosting Cloud-native apps that require rapid scaling.

Kubernetes also assists with workload portability and load balancing by letting you move applications without redesigning them.

Kubernetes components	Description
Cluster	• A group of nodes, with at least one master node and several worker nodes.
Master	• The machine that controls Kubernetes nodes. This is where all task assignments originate.
Kubelet	This service runs on nodes and reads the container manifests and ensures the defined containers are started and running.
Pod	• A group of one or more containers deployed to a single node. All containers in a pod share an IP address, IPC, hostname, and other resources.

How does container orchestration work?

When you use a container orchestration tool, such as Kubernetes, you will **describe the configuration of an application** using either a YAML or JSON file. The configuration file tells the configuration management tool where to find the container images, how to establish a network, and where to store logs.

When deploying a new container, the container management tool automatically schedules the deployment to a cluster and finds the right host, considering any defined requirements or restrictions. The orchestration tool then manages the container's lifecycle based on the specifications that were determined in the compose file.

You can use Kubernetes patterns[53] to manage the configuration, lifecyle, and scale of container-based applications and services. These repeatable patterns are the tools needed by a Kubernetes developer to build complete systems.

Container orchestration can be used in any environment that runs containers, including on-premise servers and public Cloud or private Cloud environments.

6. Cloud security

Cloud security is not limited to secure the infrastructure against attacks, it covers securing Cloud again configuration error, appliance to national and international regulations regarding the privacy of data and users Identity. It will be rater naïve to try to push the Cloud privacy to the providers' side. In practice, Cloud security is shared between infrastructure providers, the service providers, and the user of the Cloud. It is only when security requirements at the three levels (infrastructure, service, and users) are satisfied that we can talk about a secure Cloud environment (even if 100% security does not exist). Provider based security model involves SLA between provider and user defines the provider responsibility and guarantees. Data protection is attributed to user responsibility. Providers undergo certification of their Cloud infrastructure insufficient for highly distributed and virtualized environment) Customer/User must trust Provider

The Cloud responsibility diagram we showed many times in the lectures defines the responsibility between the user and the provider for the three major offerings of the Cloud (IaaS, Pass, and SaaS). The security aspects are part of the responsibilities of each role. In the version of the reasonability diagram presented in the security lecture it is clear the data is never the responsibility of the providers in all three Cloud offerings. Data owners take the responsibility to secure their data by following security best practices[54] and purchasing/lease security services offered by their providers like identify management and access control[55,56]. Cloud provider provides the **basic blocks** to build your secure services but the providers do not enforce anything. The problem of selecting and combining these **basic blocks**[57] it too complicated and require knowledge not always available on the consumer side. Certain providers provide **advanced Services** that can help Clouds users to assess and configure the security at the application level, like the AWS **Trusted Advisor,** which can help to optimize Cost optimization (idle instances, unassociated elastic IP addresses), Security (improve security of your applications by closing gaps, enabling various AWS security features and reviewing your permissions), and Fault tolerance (Increase the availability and redundancy of your applications by taking advantage of auto scaling, health checks. Multi AZ, backup capabilities). Certain providers provide advanced Services that can help **Layered Network Defense for Virtual Private Cloud** (VPC), you can use a network address translation or Nat instance in a public

---

[53] **E-BOOK** Kubernetes Patterns: Reusable elements for designing Cloud-native applications https://www.redhat.com/en/engage/kubernetes-containers-architecture-s-201910240918

[54] D.Todorov , Y.Ozkan . AWS Security Best Practices. November 2013 https://doc.uments.com/g-aws-security-best-practices.pdf

[55] https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

[56] https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis

[57] Source: Cloud Security Fundamentals | Cloud Computing Tutorial | Simplilearn https://www.youtube.com/watch?v=M_9hRPVH5SA

subnet in your VPC to enable instance in the private subnet to initiate **outbound traffic** to the internet or other ASW service, but prevent the instance from receiving

inbound traffic initiated by someone on the internet (very similar to HPC standard configuration)

In the same way, we used to "trust" the local systems administrators are doing good job in securing our local computing infrastructure, we have to somehow trust that Cloud providers have extensive expertise in security and they are applying state of the art and well-established security techniques, protocols, tools to ensure the security of the entire Cloud platform. Because Cloud systems relay on the modern paradigm of remote distributed services and online digital content provisioning, which makes security and trust relations between user and provider more complex. From the previous lectures, it is clear that separation, isolation, and security is at the art of the technology used to build Cloud systems, however it would be naïve to think that this will make the whole system secure. Consistent security must provide security at all layers correspondingly relying on trust credentials at each layer: Application –Container - Operating systems (security kernel)- Cloud platform – Network/communication – Runtime – Storage. Fundamental security challenges and main user concerns in Clouds Data security and Identity management and access control.

Security and compliance are related and, in some cases, interchangeable.

- **Security** is commonly defined as a set of technical, physical, and administrative controls in order to ensure the normal operation of a system or application. Security is often associated with the CIA triad **Confidentiality**, **Integrity**, **Availability**. The Appropriate level of security requires organizations to take measures and comply with the **numerous security controls**
- **Compliance** is a certification or confirmation that the system or an organization meets the requirements of specified standards, established legislation, regulatory guidelines or industry best practices that can be jointly defined as compliance framework. A compliance framework can include business processes and internal controls the organization has in place to adhere to these standards and requirements. The framework should also map different requirements for internal controls and processes to eliminate redundancies. There a number of standardised regulatory requirements like NIST SP 800 53[58], ISO/IEC 15408[59], HIPAA/HITECH, NIST SP 800 144, ENISA Cloud Computing Security Risk Assessment[60] and many more. Often public Cloud providers "must" comply to a number of these standards and publish the list of standards on their corporate websites[61, 62] (the certification process is quite an expenses process and not all the provider are willing or able to get a certification).

Cloud Security Alliance (CSA) Security Guidance[63], defines a number of compliance procedure and standards for Cloud providers:

[58] https://nvd.nist.gov/800-53
[59] https://www.iso.org/standard/50341.html
[60] https://www.enisa.europa.eu/publications/Cloud-computing-risk-assessment
[61] https://aws.amazon.com/compliance/
[62] https://docs.microsoft.com/en-us/microsoft-365/compliance/offering-home?view=o365-worldwide
[63] https://csrc.nist.gov/publications/detail/sp/800-144/final

- A Complete Cloud Security Governance, Risk, and Compliance (GRC) Stack is provided by the CSA. The GRC Stack provides a toolkit for enterprises, Cloud providers, security solution providers, IT auditors and other stakeholders to assess both private and public Clouds against industry established best practices, standards and critical compliance requirements[64].
- STAR Compliance levels[65]
- A public registry where you can check the compliance of Cloud providers[66].

7. Cloud Standards

In the previous lecture, we presented compliance as a fundamental aspect of the security in Cloud approach. Compliance is not limited to security, it also applies to other aspects of the Cloud approach, an extensive standardization effort has emerged since the very beginning of the emergence of Cloud systems aiming at standardizing the interaction among the components composing these systems. During the last decade, a number of Clouds standardizing bodies were setup. Cloud Strategy Partners, LLC organized a tutorial on Cloud Standardization, which is published as a transcript in the IEEE eLearning Library[67]. The tutorial looks at a couple of "standards" that apply Cloud architecture, Cloud Architecture components

Standardization bodies	Working Groups / Standards
NIST	Cloud Computing collaboration • NIST on Cloud - Standards Acceleration to Jumpstart **Adoption** of Cloud Computing (**SAJACC**) • NIST Cloud Computing **Reference Architecture** and **Taxonomy (CCRA)** [68] • NIST Cloud **Security** • NIST Cloud **Business** Use cases
ITU-T (Telecommunication Standardization Sector)	Focus Group on Cloud Computing [Established in 2010-02; Concluded in 2011-12] [69] published report consisting of 7 parts • **Part 1**: Introduction to the Cloud ecosystem: definitions, taxonomies, use cases and high-level requirements • **Part 2**: Functional Requirements and Reference Architecture • **Part 3**: Requirements and framework architecture of Cloud Infrastructure • **Part 4**: Cloud Resource Management Gap Analysis • **Part 5**: Cloud security • **Part 6**: Overview of SDOs involved in Cloud Computing • **Part 7**: Benefits from telecommunication perspectives
Storage Networking Industry Association (SNIA)	• Cloud storage[70] Cloud Data Management Interface (**CDMI**) [71]

[64] https://Cloudsecurityalliance.org/research/grc-stack/
[65] https://Cloudsecurityalliance.org/star/#star_m
[66] https://Cloudsecurityalliance.org/star/#_registry
[67] https://Cloudcomputing.ieee.org/images/files/education/studygroup/Cloud_Standardization.pdf
[68] http://collaborate.nist.gov/twiki-Cloud-computing/bin/view/CloudComputing/WebHome
[69] https://www.itu.int/en/ITU-T/focusgroups/Cloud/Pages/default.aspx
[70] http://www.snia.org/Cloud
[71] http://www.snia.org/tech_activities/standards/curr_standards/cdmi/CDMI_SNIA_Architecture_v1.0.pdf

Distributed Management Task Force (DMTF)	• Cloud Interoperability - DSP-IS0101 • Architecture for Managing Clouds - DSP-IS0102 • Use Cases and Interactions managing Clouds - DSP-IS0103 • Open Virtualization Format (**OVF**) - DSP0243
Global Inter-Cloud Technology Forum	• Use Cases and Functional Requirements for Inter-Cloud Computing,
ITU-T Focus Group on Cloud Computing Telecommunication Standardization Sector	Published report consisting of 7 parts • Part 1: Introduction to the Cloud ecosystem: definitions, taxonomies, use cases and high-level requirements • Part 2: Functional Requirements and Reference Architecture • Part 3: Requirements and framework architecture of Cloud Infrastructure • Part 4: Cloud Resource Management Gap Analysis • Part 5: Cloud security • Part 6: Overview of SDOs involved in Cloud Computing • Part 7: Benefits from telecommunication perspectives
Open Data Center Alliance	• Cloud-aware applications[72]
OGF – Open Grid Forum	• Open Cloud Computing Interface (**OCCI**)[73] • Infrastructure Services On-Demand (ISOD-RG)
OASIS	• Identity Management for Cloud • Topology and Orchestration Specification for Cloud Applications (**TOSCA**)[74]
IEEE -	WGs on InterCloud issues and Cloud Profiles • InterCloud WG (ICWG) Working Group - IEEE ICWG/2302 WG - [Active][75] • Cloud Profiles WG (CPWG) Working - CPWG/2301 WG - [Active][76]

8. DevOps[77]

Today's competition, rapidly changing regulatory and legislative requirements, ever-changing customer needs, new digital technologies. They revolutionize the way how organizations are organized and run – it forces them to be fast, adaptive and innovative, while increasing the quality, stability and security of IT at the same time. Not only within IT, but also between the business, IT departments, customers and partners. This movement is what we call DevOps, and if we start doing this we are able to release customer value faster and more frequently with higher quality. Organizations need to: (1) adopt new ways of working and operating models, (2) embrace new skills and technologies, agility and collaboration, (3) implement a radical new operating model.

DevOps defines as a culture and way of working that emphasizes collaboration between Business, Software Development and IT Operations. DevOps extends the **Agile principles** by

[72] https://www.opendatacenteralliance.org
[73] http://occi-wg.org/doku.php?id=start
[74] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
[75] https://standards.ieee.org/develop/wg/IC.html
[76] https://standards.ieee.org/develop/wg/CPWG-2301.html
[77] This section summarizes the main point of the guest lecture given by Awday Corde, former MSc student of the Joint UvA/VU CS program and now working in the banking sector

further streamlining and automating the product lifecycle and enabling cross-functional teams to take ownership of their product from an end-to-end perspective.

DevOps is not just about putting new structures and technologies in place. DevOps includes the way you think about things, the way you approach change, and the way you essentially work. Having these fundamentals right will increase your chances of success. But don't be afraid to make mistakes – be open for failure, as long as you learn. These fundamentals will be explained in the next slides, though we mainly focus on the 'hard' site of DevOps: the way teams operate. We do this by providing the practices, or building blocks, that teams can work on in their journey.

Therefore, we actually need to look at the Agile mindset, which focuses on satisfying the customer needs as fast and frequent as possible, in close collaboration with al people and parties involved. To be able to do that in a sustainable pace, we need:
A DevOps culture and mindset incorporates all of the above. But, besides that it also includes the attention of the whole teams (including Product Owner and Scrum Master) on the smooth operation and use of the software that has been and is being developed:

Agile mindset	DevOps culture
A positive attitude (solve, don't complain)	• The ability and willingness to pick up and solve incidents at the time they occur
Willingness to take ownership of the solution/service	• Constant attention to quality, deploy-ability and continuity of the software
Eagerness to learn and willingness to fail	Feeling responsible for the entire solution
A drive for success of the team with the least amount of effort	• Focus on actually solving the problem over creating 'fancy' solutions

Traditional, Linear approaches to transformation start by identifying the as is state and the to be state and then manage the transition between these two states as a long-term project, leaving no room for flexibility. This approach to transformation is fundamentally at odds with DevOps which is a journey rather than a destination. In the context of DevOps, the correct state in the future is simply unknowable and hard to predict up front. In order to bring DevOps to life you should work towards a vision by taking a cyclic, empirical approach to improve, and by changing what is needed first. This puts emphasis on learning and flexibility at the same time, while keeping transparency and control over the journey.

Way of operating - Building blocks

Building block	Definition
Team setup	• Multi-skilled engineers and team typologies that encourage thinking across silos and enable Team's to take E2E responsibility
Working practices	• You might have deployed amazing technology, but because you haven't changed your working practices – the way that work is done – you haven't actually diminished your limitations

Enabling technologies	Continuous and automated testing, integration and delivery, together with digital operations, significantly increases speed, quality and visibility
IT Landscape	• Practices and principles to build and configure your solution stack that is ready for the future

9. Cloud-Native

Cloud-native is an approach to building and running applications that exploits the advantages of the Cloud computing delivery model "Cloud-Native" is about *how* applications are created and deployed, not where. Cloud-Native help to bring the **patterns**, **practices** and **technologies**, developed by companies born in the Cloud era to enterprise[78]. A fundamental principle of a Cloud-native approach is to decompose software into smaller more manageable pieces (utilizing a microservice architecture). Cloud-native is a design pattern that strives to structure an application as a collection of loosely coupled stateless services and stateful backing services[79].

Cloud-Native can be described as an amalgamation of best practices that have been seen from companies such as Netflix, Twitter, Alibaba, Uber, Facebook and alike. Practices include, but are not limited to, continuous deployment, containers and microservices to help achieve the elastic scaling capabilities, speed of introducing new functionality and increased automation needed to cater for an unpredictable competitive landscape. So, the overall goal is to be able to adapt, and adapt quickly and cost-efficiently.

The technologies on which the Cloud-Native is "currently" based are: on containers virtualization (Docker, Singularity, …), and orchestration (K8, docker swarms, …). The benefits of the Cloud-native Approach are: (1) Self-managing infrastructure through automation, (2) Reliable infrastructure and application, (3) Deeper insights into complex applications, (3) Security, (4) More efficient use of resources.

In 2017, the Cloud-Native Computing Foundation (CNCF) [80] was created to help building a "sustainable ecosystems and fosters a community around a constellation of high-quality projects that orchestrate containers as part of a microservices architecture". CNCF tries to define a Cloud-Native reference Architecture as a four-layer architecture covering the infrastructure, the provisioning, the runtime, and the orchestration. CNCF aims to identify the ecosystems and fosters a community around a constellation of high-quality projects along the Cloud-Native reference Architecture stack CNCF role is in open source community is to foster the growth of the ecosystem, promote the technologies, and make the technology accessible and reliable (more details in the CNCF charter[81]). CNCF has established a technical Oversight Committee (TOC) who is responsible of open source project evaluation[82]. CNCF offers the following service to the committee.

<Agnosticity>

[78] Craig McLuckie, founder and CEO of Heptio, and one of the inventors of open source Kubernetes
[79] Ericsson | Cloud-Native design for telecom applications
[80] https://www.cncf.io
[81] https://github.com/cncf/foundation/blob/master/charter.md
[82] https://github.com/cncf/toc

- a Certified Kubernetes Conformance Program. Most of the world's leading enterprise software vendors and Cloud computing providers have **Certified Kubernetes** offerings[83]
- a maturity level assessment of open source project: CNCF categories the Cloud-native technology into three levels: Sanbox (entry point for early stage projects), incubating, graduated[84]. CNCF uses these maturity levels to indicate to enterprises the degree of project readiness for enterprise adoption. In 2019, the following projects advanced to "graduated" status: Fluentd, CoreDNS, containerd, Jaeger, and Vitess. This increased the total of graduated projects from two to six. During 2019, CloudEvents, Falco, and OPA joined our 14 incubating projects
- Trainings, and a certification for Kubernetes Administrators, Application developers, and service providers.
- Organize International events (conferences, Webinars, …)[85]

10. User vs provider perspective

User perspective: Zhiming Zhao has given a guest lecture presenting a user perspective, in this guest lecture, Zhiming presented the domain scientists perspective regarding of the cloud offering <TODO>

Provider perspective: we usually invite a speaker from the Dutch National computer centre (SURFsara). This year it was not possible to arrange such a talk due to the COVID-19 situation. I left the slides of last year talk in canvas for your information. I will go over the most important point of the presentation
Starting from slide 8, our guest speaker presents the services offered by SURFsara to the targeted users (Scientists), who requires complex infrastructure to solve their rathe complex problems but are not familiar with the basics of HPC and Cloud concepts like sharing, privacy, … For these users SURFsara offers compute[86] and storage services[87,88]. The computing is services are provided in various forms: HPC cluster (Cartesius[89]- Dutch National Supercomputer), GPI clusters, HPC cloud, Grid clusters, and Big data services (Hadoop). SURFsara offers IaaS Cloud services, up to 2019 SURFsara HPC cloud[90] was mainly based on OpenNebula technology, during the last two-years SURFsara started to shift towards OpenStack. SURFsara IaaS services consists of list

- List of images, Templates that could be selected form an AppMarket
- Private network for every user
- Distributed object store and file system (Ceph[91])

[83] https://www.cncf.io/certification/software-conformance/
[84] https://www.cncf.io/projects/
[85] https://www.cncf.io/community/kubecon-Cloudnativecon-events/
[86] https://www.surf.nl/en/expertises/compute-services
[87] https://www.surf.nl/en/expertises/data-storage-and-management
[88] https://www.surf.nl/en/expertises/data-and-server-services
[89] https://userinfo.surfsara.nl/systems/cartesius
[90] https://doc.hpccloud.surfsara.nl
[91] https://ceph.io

- Federation authentication (SURFcontext[92])

SURFsara is part of the national Dutch infrastructure and thus available for Researchers affiliated with a Dutch research institution and universities who can apply for limited amounts of computing time on cloud and cluster computing on HPC Cloud, Lisa, Cartesius Supercomputer and Grid directly via SURF[93]. For Large scale research projects need to apply via NWO.

Summary of the lectures

- Software architecture slide (impacted by user requirements, technology, and best practice (state of the art),
- Evolution of software architecture Monolith, service-oriented architecture, micro service-oriented architecture
- Evolution of Distributed systems (multi-clusters, Grid, Cloud)
- Abstraction (Grid) vs Virtualisation (Cloud)
- Security in Cloud a shared between user and providers
- Security is about compliance to security standards
- Not only Cloud security is standardized but the architecture, the storage, virtualisation,
- Cloud aware application development (Cloud-native), not any more porting application to cloud
- Up to now we have introduced: concepts, terminology, methods and best practices related to the Cloud approach and hopefully. We have also trained you a bit in reading "seminal" papers around the topic of Cloud systems with (White paper, conference papers, vison papers, …)
- IaaS Grid vs IaaS in Cloud? (Multi-tenant, cloudburst, reallocate,
    - What do scalability and elasticity mean in the IaaS model?
    - How **IaaS solutions** can approach **scalability/elasticity**?
    - What do availability and reliability mean in the IaaS model?
    - How **IaaS solutions** approach **availability/reliability**?
    - What do manageability and interoperability mean in the IaaS model?
    - How **IaaS solutions** approach manageability/interoperability?
    - What do performance and optimization mean in the IaaS model?
    - How **IaaS solutions can** approach performance/ optimization?
    - What do accessibility and portability mean in the IaaS model?
    - How **IaaS solutions** can approach accessibility and portability?

-

[92] https://www.surf.nl/en/surfconext-global-access-with-1-set-of-credentials
[93] https://www.surf.nl/en/apply-for-access-to-compute-services

11. Appendix – Homework summaries

1. IBM Software Migrating to a service-oriented architecture

By kishore Channabasavaiah and Kerrie Holley, IBM Global Services, and Edward M. Tuggle, Jr.,
<TODO>

2. "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision"

by C Pautasso, O Zimmermann, F Leymann
<TODO>

3. Cloud Computing and Grid Computing 360-Degree Compared

By Foster et al.:

This paper discusses the concept of Cloud Computing and what it is compared to Grid Computing. the objective of the paper is to give a definition and explain what is actually Cloud Computing. The article compares Grid Computing and Cloud Computing from **six different perspectives:**

- **Business Model**: Grid computing relies mostly on institutions joining a larger Grid to benefit from the combined compute resources offered by the aggregated resources instead of having resources idling doing nothing, it is better to aggregate those resources and share them fairly among the participating institutions. For users working at national research institutions and academia it is cheaper and easier to join a larger Grid network. Whereas for others, using Cloud and paying for only used services can be more appealing.
- **Architecture**: To enable the full potential of Grid computing, both interoperability and security are needed. This is due to the fact that compute resources used by the Grid may come from two completely different institutions with their own set of policies. The paper describes a **Grid architecture based on five layers**. Cloud systems however have way less interoperability between providers. Each Cloud provider uses standard protocols to manage their own Clouds, but there is no clear standard for interoperability across Clouds two providers simultaneous. Foster et al. (2008) explain the lack of interoperability due to the lack of business incentive for Cloud providers to invest in new protocols for interoperability. Nowadays, large Cloud providers such as Amazon or Google only provide services to migrate data between different Cloud[94], but did not develop standards to achieve at the level of computing resources interoperability. Of course, there are way ways to run applications across VM deployed on different Cloud providers, but this is left to the application developers to setup all the mechanisms for such interoperability. The most common way of interoperability is at the VM level, were most Cloud provider support the common VM image standards so

[94] https://Cloud.google.com/storage/docs/interoperability

in practice if you have a VM on Cloud provider you can run the same VM on a different Cloud provider

- **Resource Management**: another difference pointed out in the paper related to compute and data model. Foster et al. (2008) describe Grids mainly focusing on batch processing and exclusive access to computational resources, while Cloud solutions are targeted for online data processing and thus must be available to respond continuously. This could also allow latency-sensitive applications to work on the Cloud (this was not possible on Grids). Ten years later, this indeed turned out to be the case. Nowadays, it is very common to run latency-sensitive applications on the Cloud (e.g. music streaming and web services). Foster et al. (2008) make a partially wrong prediction when it comes to home hardware relating to client computing. They make a wrong prediction by stating the following: With the advances of multi-core technology, the coming decade will bring the possibilities of having a desktop supercomputer with 100s to 1000s of hardware threads/cores'. After a bit more than a decade after this article was published, these `Desktop Supercomputers with 100s to 1000s of hardware threads/cores' are still not a reality. There are CPU's with up to 64 cores and 128 threads on the market[95], but this is still far away from the `real' supercomputers
- **Programming Model**: From a programming perspective, Grid computing uses different paradigms than Cloud computing. Grid computing focuses on distributing the workload across multiple nodes and using tools such as MPI to share data with each other. The paper claims that the integration of multiple Cloud solutions is harder, as there are less resources and techniques available. One of these is the use of exposed web APIs. Today, this remains the case, where services expose an API (REST, SOAP) to communicate.
- Application Model,
- **Security Model**: Grid computing demands a higher security standard as computing resources are distributed across independent and geographically distributed institutions. The security of Cloud computing can be much simpler as all computations are executed by a single provider (single security domain). The paper further discusses that users might be unwilling to use Cloud for performing calculations on sensitive data,
- which would have to be stored in the Cloud. Today, this can be mitigated by using security compliant Cloud solutions, where the users are guaranteed a certain security level and where their data will be physically stored[96]. The mentioned technology of Google's MapReduce is still relevant today and used in the Cloud. The paper argues that the scale of data processed by Clouds is smaller than that of Grid solutions. As final advise, Foster et al. (2008) provides seven risks which one has to consider before committing to a Cloud provider.

4. Private IaaS Clouds: A Comparative Analysis of OpenNebula, CloudStack and OpenStack

By AdrianoVogel et al.

[95] https://www.amd.com/en/products/cpu/amd-ryzen-threadripper-3990x

[96] German AWS region https://aws.amazon.com/compliance/germany-data-protection/

This paper aims to make a comparative analysis of OpenNebula, OpenStack and CloudStack Cloud platforms, based on parameters flexibility, resiliency, and performance. The paper is limited to comparing the three Cloud platforms at the IaaS level. The paper extends an existing taxonomy in order to classify IaaS tools based on flexibility and resiliency. The taxonomy which originally composed of 7 layers, is extended with a management layer in order to compare the resiliency of the tools. The comparison is made while the three platforms are deployed using KVM hypervisor on a private Cloud (controlled environment) to collect new empirical insights. The performance is compared based on well-known HPC benchmarks, and scientific workloads have been used to analyze the three platforms. The authors' research results show that OpenStack is the most resilient, and it is the most flexible for deploying an IaaS private Cloud. Moreover, the performance experiments indicated some contrasts among the private IaaS Cloud instances when running intensive workloads and scientific applications