Cloud Computing in Bioinformatics: Benchmarking, Virtual Cluster, Security

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Maximilian Hanussek
aus Stuttgart

Tübingen
2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	14.08.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Oliver Kohlbacher
2. Berichterstatter:	Prof. Dr. Thomas Walter

Abstract

Questions in life sciences can be answered using bioinformatics methods. Most of the used approaches require advanced computing resources due to the complexity or the amount of data to be analyzed. Nowadays, access to powerful computing resources is easier than ever because of the widespread use of cloud computing. Especially in the field of life sciences, the interest in this technology is increasing. For example, the use of modern sequencing equipment generates data in the tera- and petabyte range that can not be analyzed with standard desktop computers anymore. The purchase of high-performance compute hardware is expensive in many aspects. In addition to the acquisition costs, there are also costs for maintenance and operation, which can also include personnel costs. In contrast, using a cloud solution can be a cheaper alternative. From a cloud user's perspective, cloud resources may look endlessly, but this is not the case. Since not all cloud users are familiar with the modern paradigm of cloud computing yet, difficulties can arise in efficient resource management and data security. Especially sensitive data as in the field of personalized medicine, require special protection, which is difficult to achieve without specific knowledge. Besides data security, resource utilization is an important issue. As the efficiency of an application has an impact on the resource usage it is advisable to evaluate an application regarding their resource sweet spot to avoid wasting resources. The evaluation can be done with the help of benchmarking tools. However, most tools in this area are not tailored to bioinformatics applications or are not very user-friendly. The same applies to the use of cloud resources as virtual compute clusters. The solutions available vary depending on the cloud platform and usually require expert knowledge.

This thesis presents new tools and concepts that enable cloud users to use cloud resources efficiently and without special prior knowledge. First, the implemented benchmark suite BOOTABLE, specifically tailored to bioinformatics applications is described. Second, VALET, a tool for automated creation and scaling of virtual clusters, is presented. Both tools were used to evaluate the scalability of bioinformatics applications as well as the saving potential of virtual cluster resources using a load based scaling approach. Likewise, a general security concept for a secured analysis of sensitive data is presented and its effectiveness evaluated against known threat scenarios. Furthermore, experiences regarding a certification process in the field of IT security are presented with a focus on the selection of the appropriate standard as well as the implementation in an academic environment with a small number of employees.

Zusammenfassung

Fragestellungen aus den Lebenswissenschaften lassen sich mittels bioinformatischer Methoden beantworten. Meist erfordert die Problemlösung aufgrund der Komplexität oder der Menge der Daten fortgeschrittene Rechenressourcen. Heutzutage ist der Zugang zu kraftvollen Rechenressourcen durch die weite Verbreitung des Cloud Computings einfacher denn je. Speziell im Bereich der Lebenswissenschaften steigt das Interesse an dieser Technologie. Beispielsweise entstehen durch die Nutzung moderner Sequenzierungsgeräte Daten im Tera- und Petabyte Bereich die mit handelsüblichen Desktopcomputern kaum noch zu analysieren sind. Eine eigene Anschaffung von hochperformanter Compute Hardware ist in vielerlei Hinsicht kostspielig. Neben den Anschaffungskosten kommen noch Kosten für Wartung und Betrieb, sowie Personalkosten dazu. Hier kann die Nutzung einer Cloud Lösung eine günstigere Alternative sein. Aus der Perspektive eines Cloud Nutzers können die genutzten Cloud Ressourcen unerschöpflich erscheinen, was allerdings nicht der Fall ist. Da noch nicht alle Cloud Nutzer mit dem modernen Paradigma des Cloud Computings vertraut sind, kann es zu Schwierigkeiten in der effizienten Nutzung und im Bereich der Datensicherheit kommen. Speziell sensible Daten, wie sie in der personalisierten Medizin vorzufinden sind, bedürfen eines besonderen Schutzes, welcher ohne Fachwissen schwer zu gewährleisten ist. Neben der Datensicherheit ist der Ressourcenverbrauch ein wichtiges Thema. Da die Effizienz einer Anwendung einen Einfluss auf die Ressourcennutzung hat ist es ratsam eine Anwendung hinsichtlich ihres Ressourcen Sweet Spots zu untersuchen, um Ressourcenverschwendung zu vermeiden. Eine solche Evaluierung kann mit Hilfe von Benchmark Tools durchgeführt werden. Allerdings sind die meisten Tools in diesem Bereich nicht auf bioinformatische Applikationen zugeschnitten und wenn ja, nicht sehr nutzerfreundlich. Ebenso verhält es sich mit der Nutzung von Cloud Ressourcen als virtuelles Compute Cluster, die angebotenen Lösungen variieren je nach Cloud Plattform und erfordern meist Expertenwissen.

In dieser Arbeit werden neue Tools und Konzepte vorgestellt, die es Cloud Nutzern ermöglicht Cloud Ressourcen effizient und ohne spezielles Vorwissen zu nutzen. Zum einen wird eine speziell auf bioinformatische Applikationen zugeschnittene Benchmark Suite beschrieben. Zum anderen wird ein Tool zur automatisierten Erstellung und Skalierung von virtuellen Clustern präsentiert. Beide Tools wurden genutzt, um die Skalierbarkeit von bioinformatischen Applikationen sowie das Einsparungspotential von virtuellen Cluster Ressourcen durch eine Last basierte Skalierung zu evaluieren. Ebenso wird ein generelles Sicherheitskonzept zur abgesicherten Analyse von sensiblen Daten vorgestellt und seine Wirksamkeit anhand von bekannten Bedrohungsszenarien evaluiert. Darüber hinaus werden Erfahrungen hinsichtlich eines Zertifizierungsprozesses im Bereich der IT Sicherheit mit dem Fokus auf die Auswahl der entsprechenden

Norm sowie der Durchführung in einem akademischen Umfeld mit einer kleinen Anzahl von Mitarbeitern dargestellt.

Acknowledgments

First of all, I would like to thank Prof. Dr. Oliver Kohlbacher and Prof. Dr. Thomas Walter, for making this thesis possible. I would also like to thank Dr. Jens Krüger for the great collaboration in the context of the de.NBI project, his helpful support in any situation and everything else. Furthermore, I would like to thank everyone, who helped me with fruitful coffee break discussions and proofreading. Finally, I would like to thank my family for their support during my whole life and particularly my beloved wife Valesca.

Contents

List of Figures

List of Tables

List of Abbreviations

2FA	Two Factor Authentication
AAI	Authentication and Authorization Infrastructure
AICPA	American Institute of Certified Public Accountants
ANSSI	Agence nationale de la sécurité des systèmes d'information
API	Application Programming Interface
BOOTABLE	BiOinfOrmatics ThreAded Benchmark tooLsuitE
BSI	Federal Office for Information Security
C5	Cloud Computing Compliance Criteria Catalog
CAIQ	Consensus Assessments Initiative Questionnaire
CCM	Cloud Controls Matrix
CIA	Confidentiality, Integrity, Availability
CPU	Central Processing Unit
CSA	Cloud Security Alliance
CWL	Common Workflow language
DaaS	Data as a Service
DDoS	Distributed Denial of Service
de.NBI	German network for bioinformatics infrastructure
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
DoS	Denial of Service
ELIXIR	European life-sciences Infrastructure for biological Information
GPU	Graphics Processing Unit
HPC	High Performance Computing
IaaS	Infrastructure as a Service
IBM	International Business Machines
IEC	International Electrotechnical Commission
ISMS	Information Security Management System
ISO	International Organization for Standardization
IT	Information Technology
JSON	Java Script Object Notation
KVM	Kernel-based Virtual Machine
LVM	Logical Volume Manager
NFS	Network File System

NIST	National Institute of Standards and Technology
NUMA	Non-Uniform Memory Access
OS	Operating System
OTP	One Time Password
OWASP	Open Web Application Security Project
PaaS	Platform as a Service
PB	Peta Byte
PBS	Portable Batch System
QEMU	Quick Emulator
QCOW2	QEMU Copy On Write 2
RAM	Random-access memory
RMS	Resource management system
SaaS	Software as a Service
SHA	Secure Hashing Algorithm
Slurm	Simple Linux Utility for Resource Management
SoA	Statement of Applicability
SOC 2	Service Organization Control 2
SOP	Standard Operating Procedure
SSH	Secure Shell
STAR	Security, Trust, Assurance, and Risk
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of privilege
TB	Tera Byte
TORQUE	Terascale Open-source Resource and QUEue Manager
TSC	Trust Services Criteria
UNICORE	Uniform Interface to Computing Resources
VALET	Virtual UNICORE Cluster
VLAN	Virtual Local Area Network
VM	Virtual Machine
XML	Extensible Markup Language
ZFS	Zettabyte File System

Chapter 1

Introduction

Today's sequencing technologies are becoming more and more sophisticated and produce larger amounts of data on the scale of tera- and petabytes in almost every -omics area (genomics, transcriptomics, proteomics, metabolomics). In order to analyze such huge amounts of data on a large scale, advanced algorithms and applications developed by bioinformaticians and others are becoming more and more important to answer the underlying scientific questions. Smart algorithms and their efficient implementation are one part of a solution. But also are the required resources to analyze large amounts of data. Algorithms in bioinformatics mostly try to solve NP-Hard problems [1]. This problem type shows an exponential increase in its runtime with the increase of the input size. To solve such a problem or to find sufficient approximations large computing resources are required. There are many more topics apart from the area of sequencing analysis that would also benefit from larger compute resources, like protein structure prediction, molecular docking, molecular dynamic simulation or, currently one of the most trending topics, machine learning. Almost everywhere an easy access to high performance computing resources is required. Access to resources in the context of cloud computing can be gained by commercial public computing cloud providers like Amazon, Google or Microsoft [2, 3, 4]. Academic cloud providers are, for example, the de.NBI cloud [5], bwCloud [6], or a hybrid one like Helix Nebula [7]. The provided resources can be categorized into various service levels, for example IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service), DaaS (Data as a Service), there exist many more but these are the most known ones [8, 9, 10]. Cloud environments usually offer high flexibility and customization abilities as the infrastructure model is based on virtualization technologies. This means that users do not get direct access to the provided hardware like for high performance computing (HPC) clusters [11, 12]. Instead, virtual machines (VMs) or software containers (Docker [13], Singularity [14] are provided, including access to storage and network services. The additional virtualization layer between the hardware and the resources (VMs, containers), accessible by users, is decoupled to a certain extent, which brings advantages in terms of resource utilization and resource management [15].

But no matter which of the providers or a custom build solution is preferred, compute resources are valuable and these resources are usually shared with others. Therefore, an efficient utilization is necessary to optimize the spent time and costs. To use offered resources efficiently, applications must be efficient as well. Some applications can benefit from multiple CPU cores due to their underlying algorithms or implementation, others can not [16]. This gives rise to two different views, on the one hand the perspective of users, which want to use as many resources as necessary. On the other hand, the view of cloud providers, which want to distribute and manage their available resources efficiently to satisfy as many users as possible with as little effort as necessary. Therefore, it would be desirable for a cloud provider to know in advance what kind of hardware would be favorable to satisfy the needs of expected users and do not waste resources related to unsuitable hardware. However, it would also be beneficial if users could estimate in advance how much resources (CPU cores, memory, storage) are required to run applications in their optimal range in terms of resource consumption and runtime. Also, throughout the development stage of an application, software developers should keep an eye on the resource consumption to find any bottlenecks. A coordination between cloud providers and cloud users would accordingly help both sides. Providers would be able to provide and manage suitable resources in a cost-efficient manner, while users would benefit from suitable hardware in terms of performance and capacity, for example in the area of storage. Therefore, knowledge of required resources is of importance.

Resource management is not only important since the beginning of cloud computing, resource optimization and management are relevant since compute resources were made available as HPC (High Performance Computing) clusters, for example [17, 18]. Both environments are providing computing resources to users, but differ in their approaches of resource access as mentioned above. Therefore, it might take time and effort for a user to change from one environment to the other [19, 20]. Reasons to switch from a physical cluster to a cloud environment can be versatile: changing resource requirements, incompatible software or other inappropriate scenarios such as operating a web service. The transfer of applications to a cloud environment can be simple, not so simple or sometimes not possible if the used applications is tuned to use a cluster environment. Cluster environments are usually not available by default. But due to the flexibility of a cloud environment, it is possible to construct a virtual cluster based on VMs and other available components, like network and storage [21]. From a cloud provider's point of view, the question arises, how the usage of provided resources can be optimized accordingly. In a physical cluster, all nodes are usually available all the time. In some cases, nodes are shut down when the load is too low in order to save energy, but this is not always the case [22]. In case of a virtual cluster, it would be possible to shut down or delete nodes (VMs) of a cluster and thus make the resources available to other users when they are not needed. To make this scaling process user-friendly and convenient, it should be dynamic and automated.

A main issue that affects a cloud environment on a technical level, as well as on the user level, is security [23, 24]. Cloud computing opens up the possibility of easy access to powerful compute resources, but can also lead to data theft by cyber criminals if the environment is not configured and used properly [25]. This is especially the case in the context of sensitive data. The possibilities offered by public clouds today are attracting a broader audience, including public sector entities such as hospitals or medical research institutions, that are responsible for larger amounts of data and want to analyze them, for example in the field of personalized medicine [26, 27]. In some cases, however, it is not possible to process the existing volumes of data without a larger IT infrastructure, but the time and costs for such an infrastructure can not be afforded. That is the point where public cloud resources can play a crucial role. But before sensitive data can be processed in a public cloud environment, a chosen cloud provider has to proof that the infrastructure and security concepts are capable of handling sensitive data and protect them according to applicable regulations [27]. Security concepts are important and should be used on all levels. On user level this starts with separation mechanisms of the individual users, that no data can be unintentionally distributed between different users. Applied measures can reach from secure data transfer and access routines up to measures at infrastructure level, which can be implemented solely by the corresponding cloud provider [28]. Also virtual cluster environments can help to process sensitive data, if a cluster environment is required, as a cluster would be exclusively accessible by certain users, having permissions for the sensitive data.

In order to evaluate a cloud provider with regard to IT security aspects and the possibility to process sensitive data on the infrastructure, certifications can be an indicator. Certifications from independent institutions can help creating trust between a user and a provider. However, there is a wide variety of certificates available on the market, which also cover different areas besides IT security [29, 30]. A more in-depth examination of the corresponding scope of the certification issued is therefore necessary from the user's point of view in order to be able to make an assessment, whether a cloud provider is suitable or not. From the provider's point of view, it is important to understand the effects of a certification, how it has to be implemented and what resources are required for it. Usually, certifications and their implementations are kept general but are mostly tailored to industrial enterprises, which are the main customers, whereas academic institutions are less in the focus due to costs and human resources [31].

The goal of this thesis is to provide tools and concepts to tackle issues mentioned above. Specifically, the topics, resource saving and resource optimization on hardware level and application level raise interesting questions and issues on the IaaS and SaaS level. The issues addressed in this thesis concern the evaluation of scaling abilities and resource consumption of applications but also the comparison of different virtual environments and their performance with each other. In connection with the topic of resource savings and optimization, interesting questions have also arisen at the PaaS level, linked to virtual cluster environments, whose solutions are also presented in this work. For all mentioned service levels (IaaS, PaaS, SaaS, DaaS) the aspect of security is already important and becomes more and more important, especially in the context of sensitive data. For all these service levels, different concepts and solutions need to be created, provided and applied to assure that sensible data are protected sufficiently. As this is a rather new area, especially in the context of public clouds operated by academic institutions in collaboration with entities from the public sector, few experience, reports and concepts are available. The experiences and concepts are presented in this thesis in order to share them and make them available to the public. To get more information about the presented issue areas, a user survey has been conducted (see Chapter 3). The developed tools, concepts and their evaluation concerning the area of resource savings are presented in Chapter 4 and 5. Developed concepts and experiences made in the context of IT security are shown in Chapter 6.

Chapter 2

Background

2.1 Cloud Computing

2.1.1 Definition

The term "cloud computing" was used by Compaq Computer, describing a paradigm, where users access services, compute resources and data via the web on remote servers for the first time in 1996. Years later, on the 9th August, 2006 Eric Schmidt (CEO, Google) introduced the term "cloud computing" to a broader audience on an industry conference, which could be seen as the point in time where cloud computing became a fixed term in the community [32]. Five years later, in September 2011, the NIST (National Institute of Standards and Technology) published its definition of cloud computing, defining it as:

> "... a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models [33]."

2.1.2 General Definition and Structure

To build and operate a compute cloud infrastructure, a set of different hardware components is necessary. A graphical overview of components and models is shown in Figure 2.1. The following descriptions are based on the guidelines for building an OpenStack driven cloud solution [34, 35], but have been generalized and are therefore valid for other cloud systems.

Loadbalancer: Depending on the complexity and the estimated access rates of a cloud infrastructure, it may be advantageous to use loadbalancers as a general entry point. With loadbalancers in place it is possible to handle network traffic in a scalable way and build a redundant and reliable setup, if two loadbalancers are used, a single point of failure can be avoided.

Controller: Controller nodes are responsible for the management of the cloud environment, hosting servers of a cloud software (OpenStack, Eucalyptus, CloudStack, Ubuntu cloud infrastructure). To build a high availability setup, a number of at least three controllers is recommendable to avoid a single point of failure and to handle upgrades with little to no down-time. A set of three is required as the metadata of a cloud infrastructure are usually organized in databases, which need to be synchronized between the different controller nodes working with quorums to avoid split databases.

Compute: Compute servers, also called hypervisors, are responsible for hosting virtual machines. Depending on the estimated workload, different sized servers should be available to share and use resources efficiently.

Network: One of the most important components is the network and the used hardware (switches, routers, network cards, cables). The network set up should be extendable, reliable and redundant to avoid single point of failures. Furthermore, it should have enough bandwidth capacities to handle the different kinds of traffic occurring in a cloud infrastructure.

Storage: A storage unit is not mandatory for a basic cloud environment, but usually needed to process and store larger amounts of data. As a storage component various open source and proprietary storage solutions are available (LVM [36, 37], GlusterFS [38], NFS [39], ZFS [40], Ceph [41], Quobyte [42], NetApp, [43] IBM [44]), but of course it needs to be checked whether they are compatible with the used cloud software.

Management: To maintain and monitor a cloud infrastructure, a separate management infrastructure is recommended. This kind of infrastructure should also be set up redundantly to avoid any single point of failures. A management infrastructure is used to install, deploy and update components or to run infrastructure wide services like DNS (Domain Name Server) and DHCP (Dynamic Host Configuration Protocol).

A cloud infrastructure built from hardware and software components, enables the following five essential characteristics mentioned in the NIST definition: [33]:

On-Demand Self Service: Applications can be deployed with less effort by users themselves without taking care of the used hardware, network configuration or other resources.

Broad Network Access: Cloud resources and the applications running on them are accessed through web-interfaces or other network protocols. If an internet connection is available, the applications and its data can be accessed from nearly everywhere. Access to a specific computer is not required. An example that illustrates this concept is Google Docs.

Resource Pooling: Cloud resources are usually not dedicated to single users. Resources are merged into larger pools and shared among users. This leads to the fact that multi-tenancy can be found everywhere in cloud environments and can be used for example to share virtual machine images with other projects (tenants) residing in the same cloud environment.

Rapid Elasticity: IT systems and requirements for it can change rapidly. Systems often have to be upgraded or scaled according to new demands. Within a cloud environment and due to the used virtualization technologies such changes can be performed easily, mostly without any down-times or service interruptions.

Measured Service: The usage of cloud resources like servers, storage, and network is monitored by a cloud provider to charge users for the consumed resources. This provides transparency for providers and users. In addition, users can monitor their resources and scale them according to their needs. A well-known example for this is AWS' CloudWatch application.

In addition to the five characteristics of a cloud environment, the definition mentions the three basic service models IaaS, PaaS, SaaS [33] which can be extended by DaaS.

IaaS: Infrastructure as a Service describes a model where compute resources are accessible for a user down to the technical levels (computing, networks, storage). Users are able to deploy compute, storage, network and other resources and run any kind of software on these resources, but have no control or access to the hardware layer of the cloud infrastructure.

PaaS: Platform as a Service builds on top of IaaS. This service provides resources for users with already supported items by a cloud provider like databases, programming languages, libraries, services and tools, an example is the LAMP (Linux, Apache, MySQL, PHP) stack. PaaS is directed to developers to build their own application on top of a provided platform without the responsibility to upgrade and maintain underlying software.

SaaS: Software as a Service provides access to applications installed and operated by a cloud provider, accountable user or IT department. Typically, these are end user applications, accessible through client devices and interfaces like web browsers or a program interface (e.g. Microsoft Office 365, Google Docs). Users do not manage any components of the underlying cloud infrastructure on the virtual abstraction layer, except for user specific application configuration settings.

DaaS: Data as a Service is not included within the NIST definition as not all cloud providers offer or have to offer this kind of service because it depends on the demands of the users. DaaS can be described as a service, where a cloud provider supplies and maintains datasets and databases directly integrated into the resources of the corresponding cloud infrastructure. Cloud environments dedicated to a special community, for example the life sciences, might benefit from standard datasets provided by a cloud provider to save storage capacities and network bandwidth as the data can be shared via the internal network connections.

Beside the various service models, there are also various deployment models that describe the accessibility of a cloud environment [33].

Public cloud: A public cloud can be defined as a cloud infrastructure available to the public, owned and operated by a business, academic or government organization, with the application of charges. Well-known cloud providers are Amazon AWS, Google Cloud and Microsoft Azure. Access to resources is usually only possible via Internet.

Community cloud: This model describes a cloud infrastructure not being completely opened to the public, but also shared by multiple organizations which are part of or support the same community, with sharing interests, aims, costs and resources. The infrastructure can be owned, managed and operated by one or more of the participating organizations or even by a third party. Further, resources can exist on or off premise.

Private cloud: A private cloud infrastructure is only accessible and operated for a single organization. It can exist on premise as well as off premise and can be managed by itself or a third party.

Hybrid cloud: This term describes a cloud infrastructure that consists of two or more deployment models (public, private, community), which are independent but based on compatible or the same technologies to allow an interoperability between them (e.g. cloud bursting).

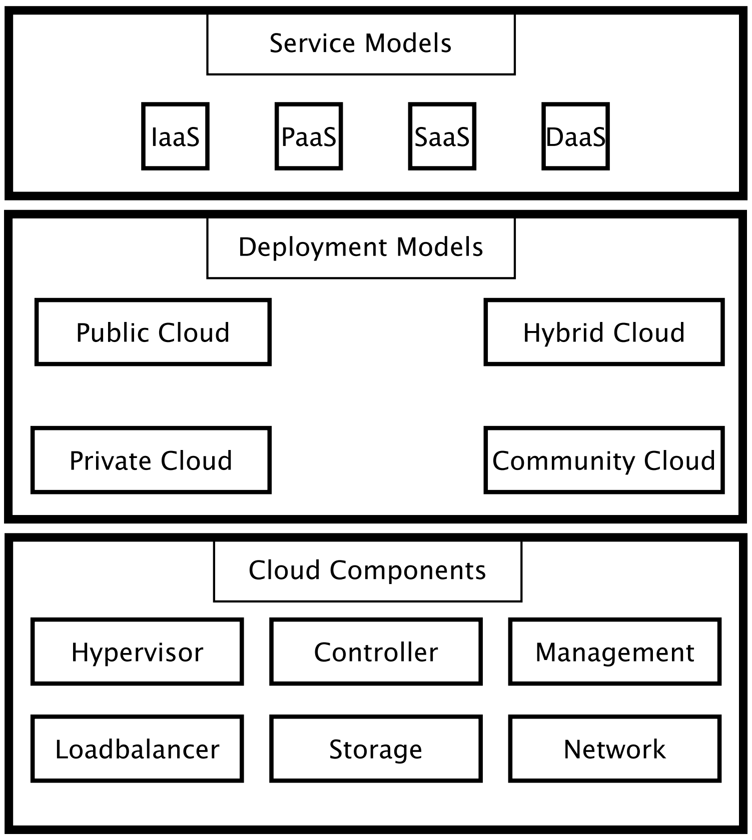


Figure 2.1: Schematic illustration of different compute cloud models and components according to the NIST cloud computing definition.

2.1.3 Virtualization Technologies

Virtualization can be divided into two major concepts, hypervisor-based virtualization and container-based virtualization [45]. Both approaches can be illustrated as different layers, starting from the hardware layer at the bottom to the application layer at the top. A comparison is illustrated in Figure 2.2. The content of this section has been taken from the original publication by Hanussek et al. [46].

The following subsections explain both technologies in more detail and show well-known representatives.

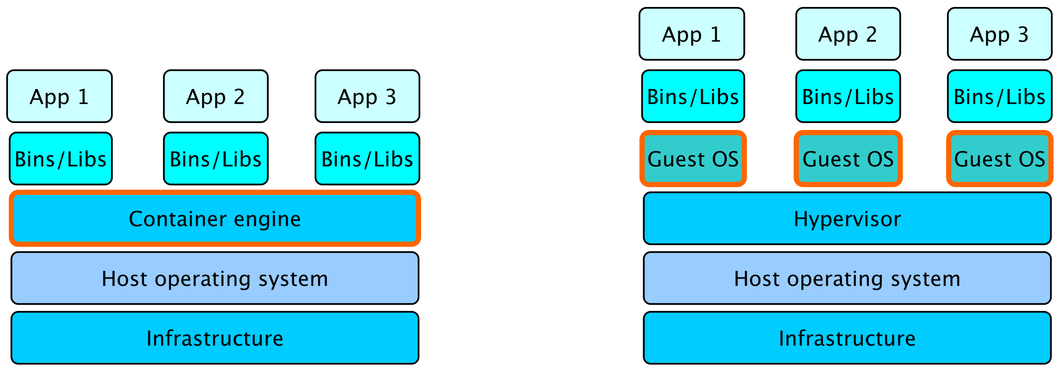


Figure 2.2: Schematic illustration of hypervisor- and container-based virtualization. The container technology concept with the container engine on top of the host operating system is shown on the left side, whereas the hypervisor technology is illustrated on the right side. The differences between the two approaches are highlighted in red.

Hypervisor-based virtualization

Well-known examples for a hypervisor-based virtualization software are VMware [47], Xen [48] or KVM (Kernel-based Virtual Machine) [49]. KVM is one of the most used options in open source environments. It is widely used, integrated directly into the upstream Linux Kernel and together with the libvirt virtualization platform the standard combination used by the cloud operating software OpenStack [50, 51]. KVM supports hardware assisted virtualization requiring a processor equipped with hardware extensions like Intel VT-x [52] or AMD-V [53]. A general aspect of hypervisor-based virtual machines is the creation of a full virtual machine with its own guest operating system (guest OS) and kernel. The hypervisor layer, sitting between the guest OS and the host operating system (host OS), translates the instructions and thus enables virtualization.

Container technologies

In contrast to a hypervisor-based virtualization approach, a container based approach is a more lightweight virtualization [45]. The most well-known representatives nowadays are Docker [13] and Singularity [14]. The lightness of container virtualization is achieved through the usage of the underlying host OS without using an intermediate hypervisor layer and a full standalone guest OS and kernel. Without these limiting factors it is possible to start a larger number of applications compared to a full virtual machine as less resources are needed. But omitting the hypervisor layer and using the host OS itself has some drawbacks. The operating system running in the container needs to fit the host

Advantages	Disadvantages
Scalability	New Paradigm
Flexibility	Non-interoperability
High availability	Dependency
Resource Pooling	Vendor lock-in
Access	Internet required
Fast deployment	Specialized technicians
Efficiency	Less control
Security	Vulnerability

Table 2.1: Non exhaustive list of advantages and disadvantages of a cloud computing infrastructure.

operating system. There are some exceptions regarding Docker containers with a Linux based operating system on a windows host system but this will not be elaborated further [54]. Another problem of containers are security concerns. As the approach of container virtualization uses the host OS more or less directly, it can be accessed through the container, which can result in permission conflicts or privilege escalations inside a container. For example, inside a Docker container, a user can get root permissions, but if a directory of the host system is mounted inside the container, which should not be accessible, it can be edited or deleted because inside the container a user has the privileged permissions [55]. Over time mechanisms have been implemented to prohibit such a behavior [56] but still the docker daemon is running as root process and can always be a potential risk. Singularity on the contrary is especially designed to run for example in HPC environments where users usually get no privileged access at all. The difference to Docker is that Singularity does not require a daemon process with root privileges. A singularity container runs with the same user ID as on the host system and therefore has the same permissions inside and outside of a container. Furthermore, containers are a lightweight solution to conserve specialized runtime environments that can be used to distribute software more easily or to increase the reproducibility of scientific workflows [57].

2.1.4 Advantages and Disadvantages of Cloud Environments

Cloud computing as modern computing paradigm differs from previous forms such as cluster or grid computing. All these paradigms have advantages but also disadvantages. The following discussion focuses on the advantages and disadvantages of cloud computing, a summary is shown in Table 2.1. Detailed information of advantages and disadvantages between cloud, cluster and grid computing and each of them can be found in the publications of Ali et al. [11], Kumar et al. [12] and Apostu et al. [58], which the following part is based on.

The largest advantages of a cloud environment are scalability and flexibility.

For a cloud user, it is possible to manage resources quickly and without contacting an employee of the responsible cloud provider. A change in resources can occur, for example, due to workload changes. According to the workload, both the number of instances used and their respective capacities (CPU cores and RAM) can be usually adapted without any downtime. For cloud providers, there are advantages on infrastructure level. A cloud infrastructure is made to be scalable and flexible. If requirements of users change, a provider is able to handle this by adding new hardware to the resource pool or changing resource limits. The possibility of resource pooling is another benefit. It enables a cloud provider to use heterogeneous hardware and merge all resources into a single pool or multiple pools. This makes it possible to procure different kind of hardware also with regard to different requirements like GPUs or larger memory. Another aspect is that users seldom produce continuous workloads, which leads to synergistic effects through the shared usage of a cloud infrastructure in terms of the resource usage. Furthermore, the underlying complexity of the infrastructure is hidden from users so they do not have to care about it. Due to the flexibility of cloud infrastructures, applications and services can be build in a high available fashion, preventing unwanted outages or down times. Over all, cloud computing can be cost- and time-efficient, no matter if handled by an organization itself or a third party. Where there are advantages, there are usually disadvantages. Cloud computing is a paradigm that is different from cluster or grid computing which means that knowledge about this topic is required. Qualified employees are necessary on both sides to fully utilize the advantages in flexibility and scalability. Especially if users or technicians change from one paradigm to the other, the learning curve can be steep. Furthermore, from a user or organizational perspective, it should be evaluated whether a complete relocation towards a single cloud environment makes sense, based on used applications and services. One issue that is of importance is the vendor lock-in. This can happen due to proprietary software and services offered by cloud providers. This means that user applications and services working with one cloud provider do not have to work with another cloud provider. Therefore, usage of proprietary elements hinders cloud interoperability. Over time, various approaches and frameworks have been created to avoid vendor lock-in issues and to improve the interoperability between cloud providers, but the problem still exists. A detailed description of this topic can be found in the publication of Bouzerzour et al. [59]. Another advantage of cloud computing is the accessibility of resources like instances, applications and data where internet access is available. But accessibility can also be a disadvantage. If an internet connection is not available or the quality of the connection is not sufficient for the required workload, resources or services are not accessible. This is specifically relevant if all applications of an organization are outsourced to a cloud environment and can only be accessed through it. This is also accompanied by a loss of control if a cloud environment is operated by a third-party provider, which makes an organization dependent on the infrastructure of the provider.

A disadvantage, that can also be an advantage, is security. If an organization outsources the operation of a cloud to a third-party, it will loose control over the system. Further, global access from the Internet leads to an increased vulnerability. However, it should be considered, that a company, whose business is the operation of a cloud environment, has appropriate knowledge and manpower in this area to protect the infrastructure and the customers data accordingly, probably even better than an organization itself whose daily business is not in the IT security sector. Therefore, outsourcing the operation of a cloud infrastructure can also increase security. All these and many more advantages and disadvantages have to be considered carefully and taken into account when thinking about whether using a cloud environment is advisable or not.

2.2 Cloud Computing in Bioinformatics

Cloud computing in bioinformatics is popular due to the demand of advanced compute resources, providing large numbers of CPU cores or RAM. Resources are required for a variety of applications, like sequence data analysis, molecular dynamics simulations, protein folding predictions and many more [60, 61]. Compute power is required because the underlying problems to solve can be complex and benefit from larger resources regarding their execution time. Furthermore, there is a need for storage in the amount of Tera Bytes and Peta Bytes, for example produced by state-of-the-art sequencing machines [61, 62]. Since advanced compute and storage hardware is expensive to purchase and maintain, the use of cloud resources provided by cloud providers is an attractive option. In addition to the required compute and storage resources, a cloud infrastructure enables the provision of services, for example through web applications or portals, which are supported by the corresponding resources in the background. The popularity of providing services via a cloud platform has steadily increased in bioinformatics in recent years and has become common practice. The characteristics of a cloud environment that are of interest, in addition to general accessibility, are its flexibility and scalability. A cloud environment makes it possible to scale up and expand resources, for example in the case of peak demands or to scale down if only few requests have to be processed to save resources and eventually costs. Likewise, the development of applications and services is increasingly taking place in cloud environments, as they are easy to access and thus enable collaborative work by sharing data and pipelines [63]. Furthermore, it is advantageous to develop services directly in the environment in which they will be provided. Additionally, platforms provided and managed by a cloud provider and the possibility to test different operating systems are further advantages. The rise in the use of cloud resources in bioinformatics as well as life sciences is made visible through the emergence of specialized cloud infrastructures, dedicated to these scientific fields. One specialized infrastructure is the de.NBI Cloud [5, 64], which provides a cloud environment tailored to the needs of bioinformaticians and their applications. Other providers active in the context of life sciences are the European Open Science Cloud (Life), Helix Nebula [7], Nectar research cloud [65], Cloud Infrastructure for Microbial Bioinformatics (CLIMB) [63] or Bionimbus Protected Data Cloud (PDC) [66].

2.3 Resource Handling and Benchmarking

To handle resources in a sensible way, benchmark applications are common tools to measure the efficiency of hardware and applications. In life sciences, this can be done using a benchmark suite focusing on bioinformatics applications. In literature, there are only a few publications in the context of bioinformatics applications [1, 67], especially regarding multithreaded applications. With BioPerf, Bader et al presented a benchmark suite to evaluate high performance computer architectures with bioinformatics applications. The benchmark suite includes ten different bioinformatics applications with very popular ones like BLAST [68], T-Coffee [69] or CLUSTALW [70]. All tools and components can be downloaded as an archive containing all the sources, pre-compiled binaries and execution scripts. Most of the input files and databases are already part of the archive, except for larger ones that have to be downloaded separately. Nearly half of the listed tools are capable of multithreading, whereas the other half is not. Further, BioPerf has a focus on technical parameters like the computer architecture design and the simulation of bioinformatics workloads. Therefore, BioPerf is valuable for hardware vendors who want to test their hardware regarding specific workloads on the architecture level, but does not seem to have that much impact on application developers as they usually want to find bottlenecks or estimate the scaling behavior of their tool. Furthermore, an extension of the BioPerf suite by own custom tools is rather complex and not designed for this purpose in the first place. Nonetheless, Bader et al. implemented a valuable tool which gives an overview about different bioinformatics applications of several topics and a large collection of suitable datasets.

Another benchmark suite for and with bioinformatics applications is BioBench, published by Albayraktaroglu et al. Compared to the BioPerf suite BioBench provides one tool less, but also focuses on the ones with high popularity like BLAST, FASTA [71], or CLUSTALW. However, the topics of the tools are directed towards sequence assembly and sequence searching but less on gene finding or protein structure prediction tools than for BioPerf. Also, the number of multithreading capable tools is smaller than for BioPerf. BioBench focuses more on benchmarking metrics on the deeper hardware level like instructions per cycle, basic block length, branch prediction accuracy, or L1 and L2 D-Cache miss rates. Like BioPerf also BioBench lacks metrics to measure scaling behavior or other resource consumption values. Also the integration of own tools is rather difficult as it needs to be integrated into the existing code base. Ten years after the publication of BioBench the second version, BioBench2, has been released on Github[1]. Compared to the first version, the tools have been changed, updated and the output has been modified. The results of the benchmark runs are presented in a tabular form, listing the wall time of each tool and a series of hardware details. Some interesting features, like automated scaling tests, integration of arbitrary tools, more information about the resource consumption and more user friendliness and convenience during the installation process are still missing.

In addition to domain specific benchmark suites like BioPerf and BioBench, more generic approaches, not specifically for the domain of bioinformatics, exist. One example is the Phoronix test suite [72]. Phoronix is a cross-platform application that provides the execution of so-called test profiles including automated installation, execution and runtime analyses. Furthermore, it has a connection to a test profile database[2] which offers more than 200 tests freely available. Throughout the research of this thesis six tests belonging to the field of bioinformatics have been found. Among them are Folding@Home GPU Benchmark, GROMACS, TensorFlow and Timed HHMer Search. Applications from the area of sequence analysis are missing. Furthermore, Phoronix offers the possibility to integrate any desired application and create own tests for it, assumed that knowledge of XML and Shell scripting is available. The supplied metrics and metadata of Phoronix are manifold, starting from power consumption, disc usage or any other data available from hardware sensors, over used compiler flags and hardware metadata. All these properties make Phoronix a valuable benchmarking tool, but it lacks a bit of user friendliness regarding the tool installation process due to the XML test description format. A drawback that arises from the domain unspecific setup are missing pre-selected and tested datasets.

For the sake of completeness, also other completely unspecific benchmark tools such as the well-known High Performance Linpack Benchmark (HPL) [73] should be mentioned. This benchmark tool is specifically designed to solve a system of linear equations which is a common task in the area of engineering. HPL measures the performance of a computing system in the unit of floating point operations. The higher the value the faster the system, regarding the procedure of solving linear equations. This property alone makes HPL not very interesting in the context of bioinformatics or even for other applications. The number of floating point operations can provide a hint on the power of a computational system concerning the application runtime, but there are many other dependencies like the demand of RAM, disc or network operations. All these are not included within a HPL benchmark.

[1] `https://github.com/reiverjohn/biobench2`
[2] `https://openbenchmarking.org`

2.3.1 Scalability

In the context of the term resource handling, performance optimizations and benchmarking the term scalability is also of importance. Scalability in general can be defined as a term that describes the ability of a system to process a growing amount of work, objects or elements in a decreasing or stable amount of time by adding additional resources to an initial system [74]. A more specific part is software scalability. Software scalability is an important topic for bioinformatics applications and software as the underlying algorithms mostly handle problems with a quadratic time complexity, or even worse, often belonging to the class of NP-Complete or also NP-Hard problems (non-deterministic polynomial-time) [75]. These problems are present in almost every area belonging to bioinformatics, like multiple sequence alignments, protein folding predictions or phylogenetic reconstructions [76]. That is why most software developers tried to make use of different parallelization strategies in order to reduce the execution time, like using multiple cores and processors on a single machine with shared memory from the very beginning. Another approach is to use multiple compute nodes connected to a compute cluster with distributed memory. Also modern programming models and implementations, like the MapReduce approach, introduced by Google [77], or Apache Spark [78] have been invented especially for the topic of big data analyses to make computations feasible in reasonable time [79].

From all these interesting strategies above the focus of this work is on the parallelization strategy using multiple cores of a single compute node with shared memory, also known as multithreading. The concept of multithreading as programming and execution model can be explained as follows, a single start process acts as a parent process and is able to spawn further threads. These threads can be executed independently of other threads which makes the parallelization possible. On systems with a single core, threads are handled by slicing and distributing the available CPU time over all running threads. On more modern compute systems with multi-core architectures, it is possible to distribute running threads over all available CPU cores so that each thread is dedicated to its own CPU core which makes it unnecessary to slice and share the available CPU time. Support for multithreading is part of most programming languages (Java, Python, C, C++), providing an API (application programming interface) to standardized thread interface implementations. Widely used APIs are POSIX Threads (Pthreads) [80] and Open Multi-Processing (OpenMP) [81]. These have already been compared to each other regarding their performance. One selected use-case even referred to the prediction of protein secondary structures [82].

2.4 Compute Cluster Environments

Computational resources are getting more and more affordable and should be accessible to everyone who needs them [83]. A large demand on computing resources mainly exists among natural sciences, but also in the humanities some applications require a huge amount of computational resources [84]. A well-established form of aggregated computing resources are compute clusters or High Performance Computing (HPC) environments. A compute cluster mainly consists of bare-metal hardware, connected to each other through specialized networking components [85]. To provide access to compute resources for users, a batch system is installed, which handles the queuing of incoming computing jobs. As stated above, HPC has been well-established over the past decades and more and more people either become familiar with it or already are familiar with it [86]. This leads to the fact, that a large number of applications are adapted and tuned to use HPC resources [87].

2.4.1 Components

The main components to operate a compute cluster besides hardware components, like compute server, storage and networking, are software components like a batch system and a file system. On top, one can use a middleware software to add additional usage capabilities, like a workflow engine or different kinds of access points. The following sections give an overview on the main components required for a compute cluster, regardless of virtual or bare metal.

File System

One of the main components of a compute cluster is a shared file system. In these days, there are lots of file systems available, like Lustre [88], GlusterFS [38], MooseFS [89], XtreemFS [90] or BeeGFS [91]. All of them are distributed shared file systems, a category of file systems that is often used for compute clusters, due to their scalability and performance. All file systems of these category are serving their underlying storage over a network infrastructure. Every file system has its advantages and disadvantages, which has been already evaluated in different publications [92, 93]. To provide insights into the structure of a cluster suitable file system, BeeGFS is explained in the following.

BeeGFS is a distributed file system, developed by ThinkParQ GmbH [91]. The working principle of BeeGFS is that it splits the object data, the effective user data, from the metadata. The metadata hold information about the object data, like file sizes and the access rights, but also information on where to find a specific file on one of the several storage servers. Overall, the BeeGFS file system consists of four components, a management server, a metadata server, a storage server, and a client server. All of these servers are single, modular software components and can be installed on different, dedicated machines or on a single machine. Generally, BeeGFS is a network file system, that connects clients and storage servers via various network protocols, like Ethernet, InfiniBand [94] or Omni-Path [95]. For more detailed information about BeeGFS see [91].

Batch System

The second important component of a compute cluster is the combination of a resource management system and a scheduler system to coordinate the distribution of different jobs to the compute nodes. There exist a lot of different batch systems and it might not be an easy way to decide which one is the most appropriate. Well known representatives are for example PBS TORQUE (Portable Batch System Terascale Open-source Resource and QUEue Manager) [96], Slurm (Simple Linux Utility for Resource Management) [97] GridEngine [98] and more [99, 100]. A batch system can be divided into two components. A resource management system and a scheduling system. The resource management system is responsible for keeping track of submitted jobs and the available resources of a cluster. The scheduler component has the task to distribute jobs, already in the queue of the queuing system, to available compute nodes. The distribution is managed by a scheduling algorithm. A scheduling algorithm has to consider different properties, such as runtime and requested resources. The goal of a scheduler is to reduce the waiting time of queued jobs as much as possible. Some batch systems come with their own, more sophisticated scheduler component, for example Slurm. Others, such as PBS TORQUE, have only a rudimentary scheduler [101] but can be functionally extended using other available schedulers like Maui [99] or Moab [102]. A more comprehensive comparison of resource management systems focused on their scheduling abilities is available from Qureshi et al. [101].

Middleware

An additional, not absolutely necessary, but useful component for a compute cluster is a middleware. A middleware in the context of HPC systems is a piece of software adding functionalities that are not directly available through the operating system or the used resource management system itself. Additional features are for example an advanced user management, a graphical user interface for the job submission or whole workflow systems and science gateways. Representatives of this kind of software are UNICORE [103], EDISON [104] and HUBzero [105]. A middleware is usually not required to run an HPC cluster but it can add some value to it, regarding the user community. Not every user that wants to use a compute cluster appreciates the beauty of the command line. Therefore, a graphical user interface, for example for the process of job submissions, can provide a more user friendly experience and make the available resources accessible to a broader audience. Especially for science gateways it is desired to hide the underlying technical system complexity and present a clean front end that handles the submission of data in a regulated and specified way so a user does not have to gain knowledge about HPC cluster environments.

2.4.2 Virtual Clusters

As already mentioned, compute clusters can be both, set up as a classical, physical (bare metal) cluster and virtual cluster using virtualization techniques, like in cloud environments. With the beginning of compute clouds, providers mostly offered them as IaaS. In the context of building a virtual cluster, that would involve a larger knowledge about shared file systems, network configurations and resource management systems. Not every user that wants to use a compute cluster has this knowledge. In the meantime, almost all larger commercial cloud providers offer own solutions to set up virtual clusters on their infrastructure. With AWS Parallel cluster [106] Amazon provides a tool to deploy virtual clusters also including a dynamic scaling regarding the utilization of the cluster. Google proposes two different strategies to deploy clusters on their Google Cloud infrastructure. The first one is a container based setup using Kubernetes [107]. The second one is a virtual cluster, based on VMs, suggesting the tool Elastic Cluster [108]. Elastic Cluster is able to deploy a cluster and also to resize a cluster manually with some limitations regarding the downsize procedure. Of course, also the Azure cloud service provided by Microsoft offers the possibility to create virtual clusters on their infrastructure with Azure Batch [109]. Furthermore, in the educational sector many different approaches can be found to create virtual clusters, mostly for Amazon Web Services (AWS) or OpenStack but also for Cloudstack- or OpenNebula-based public clouds. The work of Ruiu et al. [110] proposes a framework created for OpenNebula based clouds deploying virtual clusters including chosen user applications packed as tar balls. Another tool is BiBigrid [111] that can be used with AWS, Azure, GoogleCloud and OpenStack clouds. It offers many features including a manual up and downsizing procedure. Of further interest is the tool Wrangler from Juve et al. [112] as it deploys a virtual cluster including the workflow system Pegasus [113]. Publications focused on more special topics in the context of virtual clusters, like a fast deployment using advanced mechanisms for the provisioning of VM images [114] or the contextualization and description of a VM during a deployment [115] or using a cluster setup for larger projects as part of a science gateway [116]. Even in earlier times, there have been approaches to adapt resources of physical HPC clusters, where the technology of clouds was not that present, as the work of Chase et al. shows [117].

2.5 Security and Threats in Public Cloud Environments

The general aspect of security in clouds is important on all levels, as the high number of publications on this topic indicates. Particularly in the area of bioinformatics, sensitive data often accumulate and have to be handled and protected appropriately.

First, two perspectives should be distinguished. The user perspective and the provider perspective. Depending on the provided services (SaaS, PaaS, IaaS), the area of security of a cloud provider can differ. However, a provider is always responsible for the protection of a cloud infrastructure at the hardware level [118]. This includes the handling of access and permissions to an infrastructure on physical and software level. This can be achieved by using different policies and privilege levels of the responsible staff. State-of-the-art is the use of two factor authentication (2FA) [119]. On the physical level for example a combination of a radio frequency identification card and a personal identification number (PIN) can be used. But also bio-metric attributes are applied, like iris or fingerprint scanners [120]. Remote access on the software level can also be secured by 2FA using a hardware authentication device or one-time passwords (OTP). One of the most well-known producers for such devices is the company Yubico with its product Yubikey[3]. Further, access can be restricted to specific (internal) networks protected by firewalls, reachable through a virtual private network (VPN) connection. On infrastructure level the update and patch management of cloud hardware and software components, like operating systems and firmwares is also of importance to close security vulnerabilities [121]. If services offered by a cloud provider include PaaS and SaaS in addition to IaaS, the provider is also responsible for the security of the supported platforms, frameworks, software and applications. To support the detection and prevention of security events or incidents [122] a large number of monitoring solutions is available which are called SIEM (Security Information and Event Management). Well-known representatives on the market are Wazuh [123], QRadar [124], Splunk [125] and Darktrace [126]. The listed measures have to be carried out by the corresponding cloud provider as users of a cloud environment usually have no access to the cloud infrastructure itself. Nonetheless, users still have responsibilities in the context of security. They are responsible for their instances (VMs) regarding updates and patches for the used operating system and running software if not using a supported service offered by a cloud provider. Also, the management of firewalls and port management on VM level is part of a user's responsibility [118]. A user can decide which network ports should be opened and which should be closed. Furthermore, users are able to handle the access to their instances, for example they can change from SSH Key access to password protected access and also

[3]`https://www.yubico.com`

grant access to other persons. Therefore, the authorization and access handling is a responsibility of the users.

2.5.1 General Attack Vectors

With the increased usage of cloud computing and the advantages and disadvantages already mentioned, cloud-specific threats can be added to already existing threats in the context of IT. In order to categorize threats, different models can be used. One of them is the STRIDE threat model [127]. The acronym STRIDE stands for the six security threats of this model, explained in the following. This general model categorizes threats by their attack or result, for example unreachable services, compromised systems or loss of data made publicly available to unauthorized people. The following explanations, counter measures and content of Figure 2.3 are taken from Tabrizchi et al. [122].

- Spoofing: Concealment of origin in the context of a communication from an unknown to a known source. Used in the context of email addresses, IP addresses or DNS servers. The aim of spoofing is to gain access to personal information that allows the intrusion of a system. Spoofing attacks can be prevented trough packet filtering where header and origin are compared and rejected if they do not fit together.

- Tampering: Malicious modification of data to interfere with a system and cause a malfunction. This can be done for example by XML poisoning. A well-known type of data-tampering are ransomware attacks, which lead to unwanted data encryption by cyber criminals holding and selling the decryption key. Ransomware attacks can be recognized by file integrity monitoring. The triggered alerts and information from a monitoring system can be subsequently used to prevent further damage to the system.

- Repudiation: If a system is not able to log actions of users correctly and traceable. This circumstance can be used to modify actions of malicious users so they can not be recognized anymore. In cloud environments this kind of attack can be prevented if the integrity and origin of data can always be assured.

- Information disclosure: The leakage of information to unauthorized persons due to malicious actions. Leak of information can happen on VM level by port scanning, searching for open ports and vulnerabilities of applications running in the background. Disclosures can happen due to internal or external activities. In the case of an internal disclosure, private information are made public due to mistakes or carelessness of an employee or administrator. External disclosure attacks aim to gather information about the target system, to find vulnerabilities that can be used for further attacks. The success of information disclosure attacks can be avoided by using encryption or third-party authentication.

- Denial of Service: DoS or distributed DoS (DDoS) attacks prevent users from accessing services or resources, if successful. This kind of attack uses vulnerabilities of network protocols by transferring lots of large network packages to overcharge the network resources leading to an inaccessibility of services for non-malicious users. A measure to prevent successful DoS attacks can be a high availability setup using load balancing to distribute traffic to different data centers in different regions. Furthermore, firewalls and monitoring applications can be used to define certain thresholds for a number of incomplete connections and drop them if a threshold is reached.

- Elevation of privilege: The result of this threat type is that an unprivileged user gets privileged access and is able to compromise or destroy a system. Privilege elevation can be the result of a successful system penetration. This is a rather dangerous situation as an attacker becomes part of a trusted area. Privilege escalations can be prevented by several measures. Updates and patches that close vulnerabilities for this type of attack should be made as soon as possible. Furthermore, any policies, roles and the linked privileges to it should be audited and questioned regularly, for example if special privileges are still required. In general, the principle of least privilege should be applied.

Beneath the STRIDE threat model, Tabrizchi et al. also describe cloud specific threats that can be derived from Symantec's Cloud Security Threat Report (CSTR) from 2019, where threats of data breaches, hacked interface and application program interfaces, exploited system vulnerabilities, account hijacking, malicious insiders and denial-of-service (DoS) have been identified. In addition to the CSTR report, the information provided by the Open Web Application Security Project (OWASP) [128] have been used to categorize and identify attacks based on their attack characteristics.

Data Specific Attack Scenarios

In addition to the general attack vectors of a cloud environment, there are also special attack vectors relating to data, which is of special interest for sensitive data. According to Chin et al. [129], sensitive data can be categorized into open-access data and controlled-access data. For genomics data, a further subdivision can be made based on the level of data processing. Level 1 covers raw data, which are usually under controlled access and only accessible to users that agreed to specific data usage agreements unless researchers have the permissions to make them publicly available. An example for raw data that are open-access

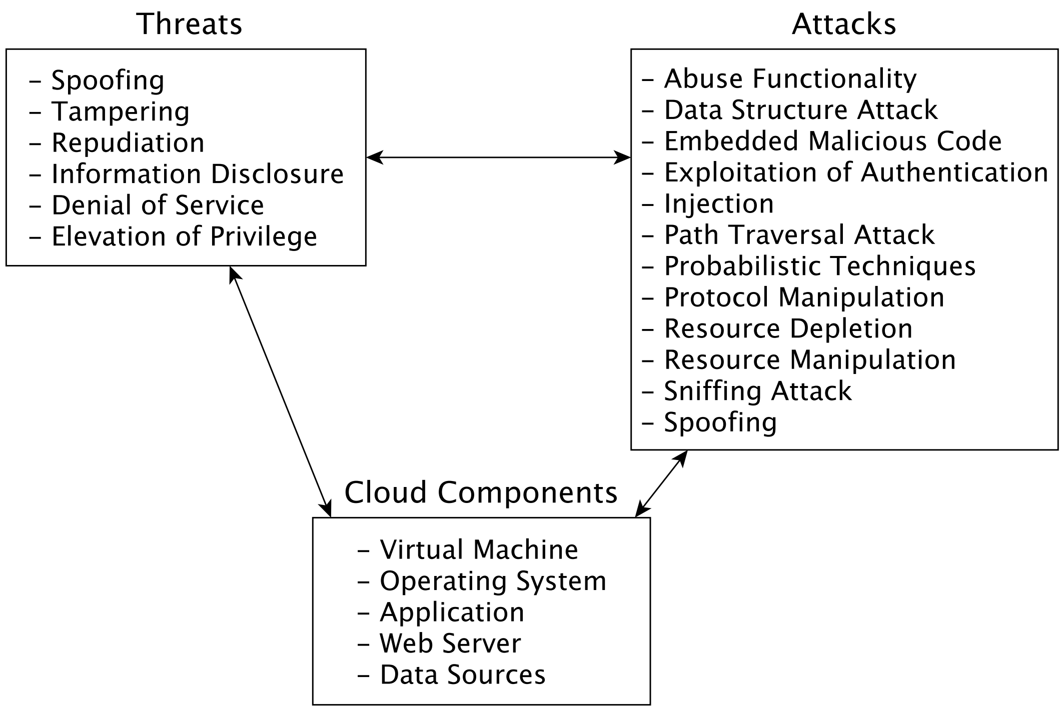


Figure 2.3: Relation of threats, cloud components and attacks in the context of the STRIDE threat model.

would be data collected by the 1000 Genomes Project [130]. Level 2 includes processed data. Data at this level can be both, open-access or under controlled access. This depends on whether individuals can be identified from the data or not. Level 3 covers interpretations of data. The same that applies to Level 2 data applies to Level 3 as well, it has to be evaluated if individuals can be identified. Level 4 handles aggregated data. From this step of processing, data can usually be categorized as open-access data as they are summarized in a way that no individual can be identified from it.

As already mentioned, the cloud computing paradigm has advantages and disadvantages, which have a direct impact on data security threats and resulting attacks [131]. A list of advantages and their resulting threats according to a cloud infrastructure characteristic is shown in Table 2.2.

To get a better understanding of data specific threats, the different states in which data can occur must be considered. Data can occur in three states. They can be at rest, in use or in transfer. For these three states different issues can apply according to confidentiality, integrity and availability (CIA) which is also a widely-used concept in the context of risk assessment [122, 131]. Confidentiality refers to the protection of data from unauthorized access. This is an important issue, especially on public but also on private clouds as cloud infrastructures

Characteristic	Advantage	Threat
Leased Infrastructure	Cost reduction	Loss of control over data
Open Infrastructure	Universal accessibility	Multiple entry points
Shared Infrastructure	Cost savings	Risk of failed user isolation
Elastic Infrastructure	Optimized resource management	Resource relocation
		Data remanence
Virtualization	Simpler infrastructure management	General risks of virtualization
	Increase in security	
Distributed Infrastructure	High availability	Loss of control over data

Table 2.2: Advantages of a cloud environment, derived from their specififc characteristics, can lead to threats. The threats are specific to data in general and sensitive data in particular.

can be decentralized and distributed geographically. Therefore, the hardware storing sensitive data does not have to belong to the owner of the data and also different laws may apply, depending on the country, where the data physically reside. Furthermore, public clouds show an increased risk for attacks compared to private clouds as they are more exposed. The term data integrity describes the protection of data from unauthorized changes. This also includes changes, that happened by accident. Changes of data can happen during the processes of creating, writing and deleting. Data integrity is a crucial point in information systems, especially in case of distributed systems. A user needs to rely on the integrity of data in order to make sure that processed analysis and interpretations are correct according to the used data. Availability does not only mean that data are simply stored. The term data availability also includes that authorized persons can access the data to read, change or do any other operations on it. The availability of data can be affected by various risks like reliability of storage components, network issues or other technical and human failures. In general, data availability and reliability in larger cloud environments might be higher than in a local infrastructure as the large providers like Amazon, Google and Microsoft have the resources to handle these risks.

Before data can be stored or used in a cloud, they have to be transferred. During a transfer all three risks for data (CIA) are applicable, data can be manipulated, stolen or corrupted. Because in this state data can travel through infrastructures, not under direct control, the risk is higher than for data at rest [132]. One solution to minimize the risk of attacks is encryption. Encryption mechanisms can be applied to data during transfer and also at rest. However, there are exceptions to be considered for data at rest. Data in the sense of archive data or backups that are rarely in use or not used can be encrypted without restricting their utilization. Data that are not actively processed but used for searches or other processes, encryption and decryption can lead to a reduced performance of these processes. In order to use data, they usually have to be unecrypted [133, 134]. There are exceptions, but their drawbacks are a high computational complexity and a performance decrease, if feasible at all [135].

Encryption can be applied in general but the state of data influences its usage. Furthermore, encryption for data in transit might not be sufficient as it can only guarantee confidentiality but not integrity [136]. Therefore, additional security mechanism should be applied [136, 137].

To summarize, cloud environments can have positive and negative impacts on the security of data. One cloud characteristic that is highly relevant is the distributed infrastructure. The location of data stored in a cloud can be intransparent for users and depending on where data are stored and the location of the cloud provider, different laws may apply than for the country of the data owners [138, 139]. This modern paradigm of data storage is especially important for sensitive data, where strict and special laws apply and vary from country to country [140, 141]

Certification Frameworks and Catalogs

After weighing advantages, disadvantages and risks, the focus is on trust towards a cloud provider, especially in the context of sensitive data. In general, the trustworthiness of a provider is difficult to measure [142]. Appropriate methods for cloud providers to proof that they are capable of handling sensitive data or protect data sufficiently are certifications. Certifications are provided by certification authorities which have the ability to verify that an organization complies with the requested requirements of a particular certification or standard. But it can be difficult to find and choose the best fitting standard as many of them are available [143]. Certifications are available for various areas like general quality management, environmental management system, health and safety standards, food safety management and especially for the IT sector. Since this thesis focuses on the area of IT infrastructures, cloud in particular, only IT specific certifications will be considered in the following. Already in the IT area, the landscape of standards is crowded [144]. The general goal of certifications and their underlying standards is to guarantee the CIA principle in terms of information [145]. The most well-known organization might be the International Organization for Standardization (ISO). Others are the Federal Office for Information Security (BSI) or cloud specific, the Cloud Security Alliance (CSA) or the Federal Risk Authorization Management Program (FedRAMP) for US Government cloud service providers. All certification organizations have their own standards and description portfolios, usually named controls. ISO offers the ISO 27001, which is a catalog of rules and guidelines to implement an information security management system (ISMS) [146], lastly updated in 2013. The BSI, a German federal office, provides the Cloud Computing Compliance Criteria Catalog (C5) [147], which is cloud specific and offers a list of controls covering the minimal requirements that should be fulfilled for secure cloud computing and is available in an updated version from 2020. CSA is an organization that reviews existing standards but also provides an own certification named STAR (Security, Trust, Assurance, and Risk) [148] available in two different levels. Furthermore, CSA provides a publicly accessible registry documenting the security and privacy controls of participating cloud providers[4].

Since the beginning of cloud computing, a large number of publications was published in the field of security and certifications. Di Giulio et al. published two papers in 2017 where different standards were compared to each other [144, 145]. In the first work, the standards FedRAMP in version 4 and ISO 27001 in the version from 2013 are compared to each other in terms of known threats in cloud computing and the question if new frameworks, like FedRAMP, are more effective than the ones created earlier. Both standards are reviewed systematically also taking into account the missing controls and their threat potential using the Cloud Control Matrix in version 3.0.1 provided by CSA. Di Giulio et al. note that there are many similarities between the two standards, but also differences. On the one hand, the distribution of FedRAMP as a standard for US authorities is significantly lower than ISO 27001, and on the other hand, the number of controls of the FedRAMP standard at a medium baseline is already twice as high as for the ISO standard. In order to investigate vulnerabilities by threats in the cloud environment, a CSA publication "The Treacherous Twelve," has been used, which identifies the 12 most relevant threats in the cloud context. From the comparison Di Giulio et al. conclude that most of the "Treacherous Twelve" are covered by both standards. For FedRAMP the authors determined gaps regarding the controls of the supply chain, access management procedures and mobile security. For ISO 27001 stricter controls for the domains network security and access to audit tools are demanded. In another study, Di Giulio et al. published a comparison of FedRAMP, ISO 27001 and the BSI C5 standards focusing on the evaluation of the effectiveness and increased protection by new standards. The analysis model is divided into two parts. One part considers attacks from the use of virtualization and the other one from outsourcing of operations. In their conclusion, ISO 27001 has been evaluated as the most efficient one among the tested standards, but according to the used analysis framework, none of them are able to guarantee complete cloud security, as insider threats can not be excluded. For FedRAMP and BSI C5 Di Giulio et al. found gaps related to the attention to training and policies to manage the workforce. For ISO 27001 gaps were found in the context of threat detection and removal.

Another study, addressing the CSA STAR catalog, has been published by Pape et al. [143]. This work examined the cloud control matrix (CCM) provided by the CSA organization and the associated Consensus Assessments Initiative Questionnaire (CAIQ). It was investigated whether it is possible to determine the corresponding security level on the basis of a CAIQ completed by a cloud provider or not. As a result Pape et al. developed an approach using an Analytical Hierarchy Process to rank the security levels of a provider based on

[4] `https://cloudsecurityalliance.org/star/registry/?view_only=trustedProviders`

customer requirements and offered capabilities of a cloud provider. This paper is of interest as it gives an overview of the CSA STAR certification and further shows that it can be difficult and time consuming for a customer to evaluate whether the offered security level is sufficient. According to Pape et al., only the information whether specific controls are fulfilled or not is not sufficient to compare different providers in reasonable time. Also the work of Rizvi et al. [149] is using the information of the CAIQ provided by the CSA organization but with a slightly different focus. Rizvi et al. propose a trust framework to review and evaluate the trustworthiness, derived from the CAIQ answers, by a third party auditor. This is of interest as the authors state, that trust is difficult to determine for a cloud customer in general and specifically if there have never been a connection between a customer and a provider. Within the implemented framework a group of auditors take the security controls from the questionnaire and rank each control in range from one to ten to determine the security level and trustworthiness of a cloud provider. Also larger cloud providers in the industry like Amazon have noticed that environments protecting sensitive data are of importance and in great demand. In this context, Bollig et al. [150] describe how they make use of the AWS Gov Cloud environment for a next generation sequencing pipeline in the area of personal medicine. To analyze the collected patient samples, larger compute power is necessary. In order to be compliant with the Clinical Laboratory Improvement Amendments [151], a secured cloud environment needs to be used. Amazon provides specific regions of their cloud infrastructure specifically for governmental purposes and other institutions in this area, being compliant with FedRAMP or ITAR (International Traffic in Arms). The work of Bollig et al. shows that institutions from the area of healthcare are already using public cloud providers and the demand will raise. The raising demands in analyzing sensitive data can be supplemented by the work of Ian Foster [118], which describes two concepts for the secure storage of sensitive data in a cloud context. Foster proposes two mechanisms, the curator model and secure enclaves. Within the curator model a trustworthy curator organization exists, that gathers sensitive data and releases data to the public or individual persons. The curator can aggregate and process the raw data to such an extent that it is no longer possible to draw conclusions about single individuals. Likewise, unique identifiers, such as names, can be removed entirely (anonymization) or replaced by identifiers that are not meaningful (pseudonymization). With anonymization and pseudonymization, however, it should be noted that this usually does not protect against the identification of individuals completely. Often, a mapping can be established with the help of additional data sets. Foster further notes that due to usage of manual steps, this approach lacks of scalability. The second presented approach is based on different security aspects that reflect potential security risks specified as safe people, safe settings, safe projects and safe outputs. Due to the control of necessary steps during a data life cycle from access and processing to resulting outputs, sensitive data can be encapsulated into a data enclave protecting data in a secured and standardized environment. Also mentioned by Foster is that larger cloud providers might have more resources and knowledge to protect data better than local institution infrastructures even by the increased risk of a public infrastructure.

The referenced publications above describe the different aspects of security frameworks and approaches to increase the level of security in cloud infrastructures. But none of them describe in more detail how to implement such frameworks, like ISO 27001, in a local organization, especially related to IT security and the academic sector. Throughout research many publications were found discussing ISO 9001, a standard for general quality management, and their impact and benefits in academia [152, 153, 154]. In the area of IT security the work of Fomin et al. [155] discusses the low adoption rate of the ISO 27001 standard. This study focused on finding reasons for the low number of publications related to ISO 27001. As possible reasons for the low adoption rate and number of publications Fomin et al. state the higher complexity compared to ISO 9001 and the higher costs in personnel and money to implement an ISMS. Another reason identified is the difficulty to measure the positive impact of an ISMS, as a successful ISMS implementation prevents damage, that would otherwise have resulted in negative costs. From this perspective, an ISMS consumes resources but does not increase revenue in a direct way, which makes it less attractive than standards that contribute more directly to higher revenues. One of the few papers dealing with the implementation of an IT standard in an academic environment was published by Muñoz et al. [156]. This work describes the adaption of the ISO 29110 basic profile, and MoProSoft, both standards for software engineering, in four different Mexican Universities. Both standards are developed specifically for smaller organizations. The Mexican MoProSoft standard for businesses up to 50 people and the basic profile of ISO 29110 for up to 25 people. The described adaption by Muñoz et al. is not directly related to an organization in general, the software development centers at the examined universities teach the ISO 29110 standard to students to close the gap between academia and industry. More interesting is that for this ISO standard different application levels exist, depending on the size of employees or the number of running projects. This would also be interesting for other standards, such as ISO 27001, in order to adapt the processes accordingly to smaller companies or to be able to give advise for their implementation.

Chapter 3

Survey on the use of Cloud Resources, Compute Clusters and IT Security

The needs and requirements of problems and questions have been evaluated by conducting a user survey in the context of the German Network for Bioinformatics Infrastructure (de.NBI). The survey has been published via the social media accounts of de.NBI. The period to answer the survey has been set to 2 months. In total 37 participants have been counted. The survey is divided into four parts focusing on different topics. The first part asks for information of general interest, for example what kind of scientific background the participants have and how they use a cloud environment like the de.NBI Cloud. The second part asks questions in the context of the scaling of applications and resource usage. Third, information about the topic of compute clusters are collected from the participants. The survey finishes with questions about security, including the topics sensitive data and certifications.

3.1 Results

The results of the general part were used to gain an overview about the scientific background of the participants and their available knowledge. The results are summarized in Table 3.1, including the most selected answers. The first question about the field of expertise was mostly answered with bioinformatics (87%) and genomics (42%). But also participants belonging to the fields of machine learning, metabolomics, molecular dynamics and chemistry have participated in this survey. Behind the category "Other" additional participants from the areas of transcriptomics, software design and computer science were found. The answers of the second question revealed that most of the participants have access to local high performance compute resources like a compute cluster but also to

Question	Most	Second	Third	Fourth
Field of expertise (33)	Bioinformatics (82%)	Genomics (42%)	Proteomics (21%)	Other (24%)
Local compute access (33)	HPC (82%)	Cloud (45%)	None (12%)	Other (3%)
Cloud usage reasons (23)	Insufficient resources (48%)	Webservices (44%)	Need of storage (26%)	Other (35%)
Cloud usage (23)	Single VMs (61%)	Kubernetes (26%)	Virtual cluster (22%)	Other (4%)
Main usage (23)	Compute jobs (74%)	Development (57%)	Services (52%)	Storage (17%)

Table 3.1: Summarizing table, showing the top answers of general questions, ordered by their percentages, multiple answers were possible for all questions. The numbers in brackets in the first column express the number of participants that answered a question.

local cloud resources. Of further interest was how the participants use available cloud resources. Most answered that they use single VMs, but also the usage of Kubernetes and virtual clusters was in the top answers. It was also of importance to find out how cloud resources are used. Most of the participants answered that they use them for compute jobs but also a high fraction uses cloud environments to develop software and to offer services or to store larger amounts of data. In summary it can be said, that the participants mostly belong to the field of bioinformatics, have experience with compute clusters and cloud environments and using single VMs as well as virtual cluster solutions. Available resources are largely utilized for compute-intensive tasks, but also for the development of applications or the provision of services.

In the subsequent block of questions, participants were asked more detailed questions regarding resource consumption of applications and their position towards tools, evaluating the scalability of applications. The results of this part are summarized in Table 3.2. The first question of the scaling topic was kept very general and asked whether the participants have already thought about scaling capabilities or not. Most of the participants (80%) have already thought about this issue but 20% have not. Therefore, it was of interest to get information about the reasons why people did not think about it. The top answers on this question were: 1.) There is no need, computations are fast enough for their purposes, 2.) The knowledge of this topic is not available or not sufficient, 3.) Participants are afraid of the complexity of this subject. The participants who have already thought about the topic of scaling were asked which resources would be most interesting for them in terms of scaling. A large majority (88%) answered the question with compute cores, followed by RAM and disk IO (both 62%) and parallelization with 31% (both, intra-node and inter-node). After the questions about the importance of specific resources, it was important to determine whether there is a general need for a tool to test scalability properties or to find bottlenecks, to which 75% of the participants said yes. To gain a deeper understanding of the features of such a tool are desirable or relevant, participants were asked to rate a list of features based on their importance to them on a scale of 1 (not at all important) to 5 (extremely important). Most importantly, were

Question	Most	Second	Third	Fourth
Thought about scaling (20)	Yes (80%)	No (20%)	—	—
Why not thought about scaling (5)	No need (60%)	Missing Knowledge (40%)	Too complex (20%)	—
Resources of interest (16)	Cores (88%)	RAM (62%)	Disk IO (62%)	Parallelization (31%)
Demand of evaluation tool (20)	Yes (75%)	No (25%)	—	—
Importance of tool features (18)	CPU information (4.28)	RAM information (4.17)	Disk IO (3.83)	Automated scaling (3.41)

Table 3.2: Results of questions belonging to the scaling part, ordered by percentages or importance, multiple answers were possible, except for decision questions.

detailed information about the CPU (4.28) as well as RAM usage (4.17), which refers to the question about interesting resources. The moderately important categories cover features such as automated scaling (3.41), disk IO information (3.83), and pre-selected test data sets (3.31). The least important feature was the comparison with already existing applications (2.94), but still close to the category of moderately interesting.

In the third part of the survey, participants were asked more detailed questions about the topic compute clusters in order to gain a better understanding of the requirements and to identify potential areas for improvement. The results of this part are summarized in Table 3.3. In general, nearly all of the participants (95%) that answered this part of the survey have or had experience with cluster environments, whereby the highest percentage (86%) has or had experience with a physical, classical compute cluster. However, the proportion of experience with a virtual cluster is also quite high with a percentage value of 45%. Continuing, the reasons why a virtual cluster is used instead of a physical one were of interest. The answer flexibility/customization was chosen most frequently with 54%, followed by incompatible software with 22%. Only a few participants (11%) were not satisfied with the waiting time and therefore preferred a virtual cluster solution. But also a large fraction of valuable answers are hidden in the category "Other". Participants listed reasons like training, teaching or redundancy in case the used physical cluster systems fails or simply to get more resources than available only by a physical cluster. Another general question to participants that do not use a virtual cluster already was whether they would use a virtual cluster if it would be easy to deploy. Most answered with Yes or Maybe (both 45%). To gain an impression of the qualities users expect from a virtual cluster, no matter if they use one or not, they were asked about the importance of certain features. Most important was an easy deployment (4.47) followed by robustness of the cluster and its components (4.32). Of high interest were also the network and file

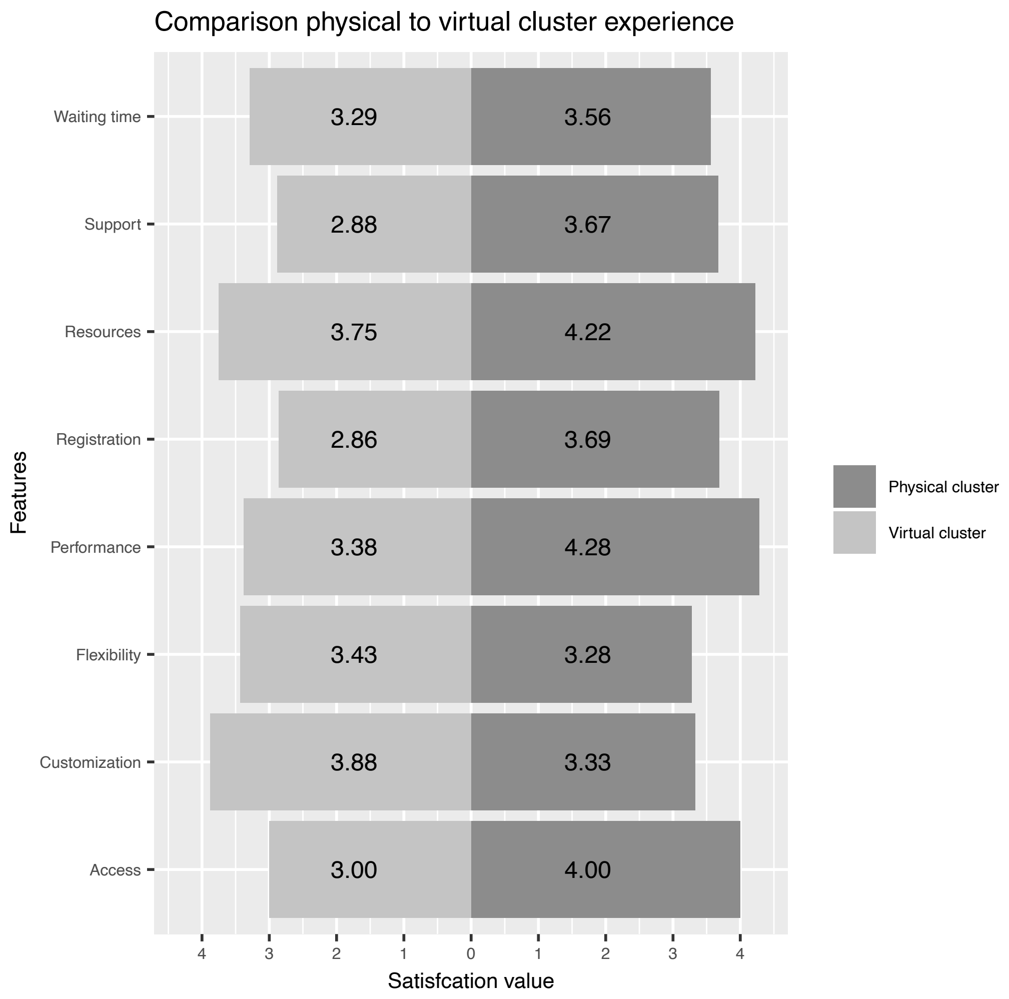
Question	Most	Second	Third	Fourth
Cluster experience (22)	Yes (95%)	No (5%)	—	—
Cluster environment (22)	physical (86%)	virtual (45%)	—	—
Why virtual (9)	flexibility and customization (56%)	software (22%)	waiting time (11%)	Other (44%)
Virtual if easy to deploy (11)	Yes (45%)	Maybe (45%)	No (10%)	—
Virtual cluster features (19)	Simple deployment (4.47)	Robustness (4.32)	Network performance (4.11)	File system performance (4.11)

Table 3.3: General questions about the participants' experience with cluster environments (virtual or physical), ordered by percentages or importance.

system performances (4.11). Downstream features are the possibility to resize a cluster (manually (3.26) or automated (3.00)), a monitoring solution (3.68) and a workflow engine (3.39).

As the survey differentiates between virtual clusters and physical clusters, the made experience regarding different topics with a physical cluster compared to a virtual cluster (see Figure 3.1) were evaluated. The results show that the average score for the topics available resources, performance, waiting time, account registration process, support and access for physical clusters are higher (more satisfied) compared to virtual clusters. For the topics flexibility and customization the scores for virtual clusters are higher compared to the values of physical clusters.

In addition to the topics of scaling and cluster environments, another relevant topic is the processing of sensitive data, for example in the context of personalized medicine and therefore the topics security and trustworthiness of a cloud provider. In order to find out how the level of trustworthiness relates to non-certified and certified cloud providers, the survey participants were asked about this. The results of this question block are summarized in Table 3.4. First of all, it was of interest whether the participants would process sensitive data in a public cloud environment or not, which resulted in a nearly fifty to fifty distribution. Afterwards, the participants were asked whether a certification would increase their confidence towards the trustworthiness of a cloud provider, which resulted in the answers "Yes" with 53%, "No" with 21% and "I do not know" with 26%. In the following question, participants were asked whether they already know the difference between two cloud related certifications (ISO 27001, BSI C5). The result was that three-fourths were not aware of the differences and only one-fourth declared that they were aware of them. Finally, the survey participants were asked how likely it would be for them to process sensitive data with a cloud provider with neither one or both of the previously presented certifications, or both certifications. The scale ranges from 1 (very unlikely) to 5 (very likely).



.

Figure 3.1: Comparison of participants' (22) experience, made with a classical, physical compute cluster and with a virtual cluster. The lowest possible value is 1 (totally unsatisfied) and the highest possible one is 5 (totally satisfied).

Question	Most	Second	Third	Fourth
Process sensitive data on cloud (19)	Yes (53%	No (47%)	—	—
Confidence increase with certification (19)	Yes (53%)	No (21%)	Do not know (26%)	—
Aware of differences between ISO 27001 and BSI C5 (19)	No (74%)	Yes (26%)	—	—
Different certifications (19)	without (2.43)	ISO 27001 (3.71)	BSI C5 (3.64)	Both (3.93)

Table 3.4: Results of questions from the security question block in percentages for the first three questions. The last one describes the likelyness to process sensitive data in cloud environemnts with different levels of certifications. The values range from 1 (very unlikely) to 5 (very likely). The last value noted as "Both", means that a cloud provider owns an ISO 27001 and a BSI C5 certification.

The answers show that it is rather unlikely that sensitive data will be processed by a cloud provider without certifications. But if only one of the certifications is granted, the trustworthiness increases by 1.23 (BSI C5) and 1.28 (ISO 27001). The participants rated the probability close to 4 (likely) if a provider would hold both certifications.

3.2 Discussion

In summary, there is a demand regarding the analysis of the consumption of compute resources as well as their scaling behavior. As users and developers are among the respondents, both are identified as potential targets for an analysis tool. The user group would be able to evaluate tools with regard to their potential scalability and resource consumption. Especially in a flexible computing cloud environment, where resources can be exchanged easily, like using SSDs instead of HDDs if disk IO is a critical resource or larger VMs with more compute cores or RAM can speed up calculations, a resource analysis might gain valuable insights and therefore advantages regarding the execution time of a specific tool. Developers can benefit of a resource analysis during the implementation of software and tools to find potential bottlenecks or critical resources and use the information to fine tune code parts and algorithms to reach a better performance. In order to meet this demand, a benchmarking tool called BOOTABLE has been developed throughout this thesis, that on the one hand already provides a selection of different well-known applications from the area of bioinformatics, including pre-selected datasets, and on the other hand also allows to include own applications and use the provided data sets just as the implemented metrics and analysis features. An in-depth description of BOOTABLE and an additional scaling study is introduced in Chapter 4.

From the resulting answers of the virtual cluster part it can be concluded, that there is a need for virtual cluster environments that are easy to deploy, robust, and have a high disk IO and network performance. To obtain information on the satisfaction level of certain features regarding physical clusters and virtual clusters, they have been compared with each other. For most of the categories the participants answered that their made experience with physical clusters are better than for virtual clusters. Noticeable is the quite large difference between the support experience and the access process. Both could be related to issues with the deployment process, which might be too complicated. This assumption is also supported by the obtained results, that not many of the participants use a virtual cluster but more would like to use one if it would be easy to deploy. The large difference in the support experience could be related to the fact that supporting staff of cloud resources see themselves more as IaaS providers and therefore not responsible for applications that belong more to the PaaS area. So these topics show a potential for improvements regarding virtual cluster environments. Furthermore, the performance has been identified as an important point, but it must be taken into account that a virtualized environment can offer at most an equivalent performance compared to a bare metal cluster environment. Usually a virtual cluster would not be able to compete due to the additional virtualization layer. Nevertheless, one should try to get as close as possible to the values of a physical cluster. The most advantageous properties of a virtual cluster are its flexibility and customization possibilities. However, the survey results of this topic show that the experience scores are not that different from each other, which points towards a larger improvement potential in this area. To meet the need of an easy-to-use deployment of a virtual cluster, a tool has been implemented for that purpose. In addition to the basic components it also includes the middleware UNICORE and its workflow engine, as well as the monitoring system Zabbix. In addition, to control resources dynamically, a meta-scheduler has been implemented, which estimates the resources based on the workload and scales them accordingly. The developed tool named VALET and an evaluation of the implemented meta-scheduler is presented in detail in Chapter 5.

The responses received regarding the security questionnaire block indicate that certifications generally can increase the trustworthiness of a cloud provider, especially with regard to the processing of sensitive data. However, the responses also showed that granted certifications tend to be a black box for users and that the scope of their effect is unclear. Therefore, a need for clarification in this area is identified. In order to bring more clarity into the area of certifications, especially in the context of IT security, this work explains how the cloud related certifications ISO 27001, BSI C5 and CSA STAR can be acquired and how they can be valued. In addition, concepts are presented. Further explanations on these topics can be found in Chapter 6.

Chapter 4

Bioinformatics Benchmarking Tool Suite

4.1 Motivation

To build and run a compute cloud a stack of hardware is required. Usually, computational hardware has to be purchased/acquired or you have the choice to use already existing hardware. The decision on the kind of hardware that has to be procured or used, should depend on the applications and use cases it will be used for. Questions can be: How robust is the underlying hardware? How fast can it deliver the application results? Is the number of compute cores more important than the clock rate? How much RAM is required? All these and many more questions need to be considered in order to purchase and use computing hardware in an efficient and cost-saving way.

To test different hardware, a test environment is required, consisting of a number of tools and suitable datasets within the scope of the respective discipline. These base elements combined lead to a benchmarking suite capable to handle required measurements. In order to get comparable results, a benchmark suite has to fulfill some constraints. First, every tested tool needs to be installed in exactly the same version and in the same way. Second, if it needs to be compiled, it must be the same compiler with the same parameters to exclude any variation and therefore deviations in the measured values. In addition to the applications, the data used to run the desired tools, has to be the same as well. Even further, the datasets have to be chosen carefully, because their size or complexity will have a direct impact on the runtime. Summarizing all these requirements above, a reproducible deployment that guarantees a minimum standard of equality is required. Furthermore, the application scenarios for a benchmark tool would not be limited only to a procurement process. Examples of application areas for a benchmarking suite are presented in the following.

Hardware Procurement Process: In order to buy and test hardware, a hardware vendor is usually contacted. The hardware vendor or reseller, in most cases, has no clue about the used applications or how to install them. Neither does he know which datasets are required or how to execute the applications. Furthermore, hardware vendors might have another opinion than a researcher about which results of a benchmark are meaningful for the performance of the used tools. Explanations on how to install and run the proposed applications and the desired results would be necessary, but this can be time-consuming.

Hardware Operator: Important for a hardware operator is the focus on a well formed utilization of the underlying hardware, so that requested resources are used wisely and not wasted and therefore prevent others from using resources. In order to give users an estimation about which amount of resources would make sense, the operator will need some measurements of the used tools. For large and commercial providers this might not be interesting as their users pay for the allocated resources but for academic and private, specialized compute centers, supporting a specific community resource utilization can be critical and it might be worth the effort to investigate the scaling behavior of used tools and applications.

Hardware Vendor: Even the hardware vendors would benefit from a benchmark suite directly. They would not have to care about how to install the tools, run the tools or find suitable datasets for them. This possibility could lead to systems optimized for specific workloads, especially for bioinformatics applications.

Application Developer: From the application developer's point of view, it would help to test tools regarding their utilized resources and scaling behavior, to see where the limits or bottlenecks are and to overcome them eventually. For example it would not make sense to increase the throughput of an application or speed up the underlying algorithm if the bottleneck is the disc read/write speed. But to reveal a limit like this, a supporting tool that performs all the necessary measurements would be helpful.

User: From a user's perspective, it is beneficial to know whether a certain application delivers results faster or not, if using more resources and if not, to find the best configuration of resources delivering a sufficient performance.

This chapter focuses on the responsible use of compute resources in a cloud environment. An analysis tool that creates reproducible benchmarks would be a supporting component with regard to the resource management as it would enable a cloud provider to select the most suitable hardware and helps cloud users to find appropriate resource parameters for their applications. Since there are hardly any usable benchmark tools for applications in the field of bioinformatics, a tool for the automated execution of benchmarks on application level, as well as the analysis of the used resources at runtime, should be implemented. For the evaluation of the benchmark tool, the integrated tools should be tested for scalability and performance with different dataset sizes in different execution environments. The presented content of the following sections has already been published within PLoS Computational Biology [16] and the Future Generation Computer Systems Journal [46].

4.2 The BOOTABLE Benchmark Suite

To enable application developers and users to explore and evaluate their applications with regard to bioinformatics workloads, the tool BOOTABLE (BiOinfOrmatics ThreAded Benchmark tooLsuitE) has been developed throughout this thesis. With BOOTABLE it is possible to get an overview or insights into the resource consumption and scaling behavior of tools and applications. The subsequent sections explain how BOOTABLE is used and what kind of implemented features are available.

4.2.1 Features

BOOTABLE includes plenty of features made to get a rapid overview about the used system, used tools and their contribution to the overall run-times and resource allocation. It is focused on applications that are capable of multithreading in almost every part of their algorithms and implementations. The reason for this is that BOOTABLE is designed to produce an appropriate load on the test-hardware and to get further insights into the scaling possibilities of the chosen tools. For both application cases, single threaded tools would be inappropriate as a scaling test would not work and in order to create a high load on the system, tools have to be started simultaneously. Furthermore, BOOTABLE is available in various installation formats.

Benchmarks can be executed without any given user input by carefully chosen default values. Of course the default values can be altered. Users can choose between different sized datasets, the number of times the same benchmark should be repeated (replicates) to minimize the impact of outliers and the number of used threads. The thread option is available as all chosen applications are capable of multithreading. Further, it is possible to choose certain tools or tool groups like genomics, molecular dynamics or machine learning that cover specific tools of these research areas.

In addition to the already integrated tools, BOOTABLE provides the possibility to use own tools while users benefit from the already implemented metadata and resource usage data collections. This is especially true in combination with the available scaling mode to test the scaling behavior of an application. All gathered data are collected and provided in a general report and multiple sub reports. All of the mentioned features are explained in more detail in the following subsections.

Installation

It can be very time consuming to download and install all desired tools and datasets. In order to overcome this obstacle BOOTABLE provides a completely automated installation process in two variations and three further installation-setup possibilities. All used tools and datasets are made available through an S3 capable storage system to avoid version-changes regarding the applications and download path-changes in terms of the relevant datasets, which has been experienced during the evaluation of the Bioperf and Biobench tools.

- Bare-metal: The first and general option is to use the provided installation tool, which will download all the tools and datasets, install them properly and place every dataset in the expected location.

- Ansible playbook: Another option to install BOOTABLE on a host is to use the supplied ansible-playbook [157]. This might be helpful if a DevOps environment based on Ansible is available, to integrate BOOTABLE more easily.

- Docker: In addition to the direct host installation, containerized versions are also available. A Docker [13] image with all tools and datasets already installed is publicly accessible from Docker Hub[5]. This option makes it possible to run BOOTABLE on hosts decoupled from public networks. Further, it can be tested whether Docker containers are a suitable runtime environment compared to a bare-metal installation with regard to losses in performance of an application.

- Singularity: The second option of providing BOOTABLE in a containerized version is Singularity [14]. This alternative to the Docker image is provided for environments that have higher security levels and do not allow to run the Docker daemon, HPC clusters for example as the Docker engine requires root/sudo access. In contrast, Singularity is designed to run on HPC clusters in the user-name-space.

- Disc image: The last offered option to run BOOTABLE is the usage of a disc image in the QCOW2 format. In this case all tools and datasets are already installed, as well. The disc image is intended for cloud environments like OpenStack [158] or other virtualization environments, where it is possible to make the disk image available and create virtual machines from it.

[5]https://hub.docker.com/r/maximilianhanussek/bootable

Installation Check

As operating systems and versions of it can vary and do not behave as the development or test system, installation processes can fail with errors or even silently. Especially in order to handle the last case, a helper tool is provided within the BOOTABLE tool suite that checks, whether every tool is installed in the expected file path and the binary files can be executed, the same holds for the datasets. The tool informs a user about the tests done and signalizes if everything is correct (colored in green) or if a check failed (colored in red). Afterwards, a user has the possibility to start a re-installation tool by tool. This option is only available for the bare-metal installation via the installation script. A user can simply execute the installation script again and it will recognize that tools and datasets are already downloaded and installed and will ask, whether or not one of the tools should be re-installed. This can become a very time saving feature as a user do not has to install or download everything from scratch as the installation time is also related to the network bandwidth.

Integrated Tools

The BOOTABLE tool suit contains various applications from widespread areas of bioinformatics or life sciences. A complete enumeration is represented in Table 4.3. The listed tools cover a wider range of bioinformatics workloads like sequence alignment, genome assembly, multiple sequence alignment but also molecular dynamics and machine learning tasks. As stated before, all tools are capable of multithreading in most parts of their implementation and can therefore be used for workloads of various sizes by adapting the number of threads. A more detailed description of the tools and their application area is presented in Chapter 4.3.

Integrated Datasets

In addition to the integrated applications listed above, a number of datasets have been chosen to run the applications on real scientific data and save the time consuming step of searching for a proper dataset with a fitting size. As the size of a dataset has a large impact on the runtime and resource consumption datasets have to be chosen with care. A list of the used and integrated datasets is provided in Table 4.5. An in-depth explanation about the used datasets can be found in Chapter 4.3.

Benchmark Execution

The core functionality of the benchmark tool is executed through the `run_benchmarks` tool. It is possible to start a benchmark run by just using

Parameter	Information
Hostname	bootable-image-creation.novalocal
Architecture	x86_64
Vendor ID	GenuineIntel
CPU Family	6
CPU Model	85
CPU Model Name	Intel(R) Xeon (R) Gold 6140 CPU
L3 Cache	16384KB
Processor Speed	2.30GHz
CPU cores	36
RAM	1,4TB
Compiler	gcc-version 4.8.5

Table 4.1: The generated system information chart provides a brief overview of the technical details of the used hardware. A user can find information about the compute architecture or which CPU model has been used or the clock rate. Aditionally, information about cache size, RAM size and the used default compiler are provided.

the default parameters which will result in a medium sized benchmark with a quite moderate runtime. A list of all available parameters and their description can be found in Appendix A. Further descriptions on how to run a benchmark can be found in the BOOTABLE GitHub repository.

Automated Report Generation

After a benchmark run is finished, different measured time metrics of the benchmark tools are available in plain text format. In order to provide a more handsome view on the data, a report generator has been implemented that collects the created runtime information and displays them in tables and graphs. The report contains information for beginners as well as for advanced users or hardware experts.

In the following, examples of a finished benchmark report and their corresponding information are shown in Table 4.1 and 4.2. The general overview report gives some first information about the used hardware, the measured time metrics (real, user, sys) of all tools for every run and also average values grouped by tools or replicates. An example report can be found in the corresponding GitHub repository in the examples directory. The enumerated basic information are mostly directed to users just starting with the topic of benchmarks and their results. Furthermore, if a scaling benchmark run has been executed, it is possible to generate so-called scaling plots that show the scaling behavior of chosen tools for different thread sizes, an example is shown in Figure 4.2.

The general information provided by the report are backed up by more detailed information addressed to more advanced users, including data about the resource consumption and the compute system itself.

In order to monitor the resource consumption, the tool nmon [159] is used

Tool	Replica 1	Replica 2	Replica 3
Bowtie2_build	515.01 / 510.19 / 4.81	508.48 / 504.01 / 4.46	514.79 / 510.26 / 4.5
Bowtie2_align	47.07 / 46.27 / 0.86	44.67 / 43.84 / 0.86	45.21 / 44.36 / 0.88
BBMap	204.45 / 340.73 / 13.45	206.73 / 340.22 / 14.39	209.36 / 347.53 / 14.79
BWA_MEM	111.89 / 111.82 / 1.17	111.33 / 110.57 / 1.89	107.54 / 107.37 / 1.31
Velveth	8.12 / 91.89 / 12.84	8.4 / 92.17 / 13.24	8.36 / 92.61 / 13.95
Velvetg	49.83 / 153.72 / 1.49	46.86 / 153.04 / 1.34	47.11 / 156.09 / 1.14
IDBA	139.5 / 137.64 / 1.86	135.85 / 134.06 / 1.79	134.71 / 132.9 / 1.81
clustalOmega	6.22 / 5.87 / 0.35	6.21 / 5.86 / 0.34	6.09 / 5.74 / 0.34
MAFFT	15.72 / 15.6 / 0.14	15.77 / 15.66 / 0.13	16.49 / 16.37 / 0.13
SINA	367.83 / 349.28 / 23.49	351.11 / 336.4 / 14.7	350.05 / 335.03 / 15.02
TensorFlow	1120.02 / 1109.79 / 23.49	1118.65 / 1108.61 / 23.48	1129.57 / 1117.74 / 25.41
GROMACS	793.55 / 793.01 / 0.34	789.42 / 788.89 / 0.34	793.72 / 793.16 / 0.36
SPAdes	2456.67 / 2432.13 / 23.81	2443.70 / 2420.04 / 23.71	2485.10 / 2460.69 / 24.50

Table 4.2: To present the collected runtime values in a convenient way, they are illustrated in a tabular format. The three measured runtime values (real runtime, user runtime, system call times) are available for every used tool and for every executed replicate. This representation gives a user a quick and clean overview of the raw time data values for deeper investigations.

and integrated into the benchmark suite. Nmon provides a rich output of metrics starting with CPU utilization, process monitoring, memory consumption metrics, network traffic and disc information about read and write access. All these metrics are collected by snapshots within an interval time of two seconds to get a detailed resolution. The information are available as interactive documents in HTML format. An example of such a graph is illustrated in Figure 4.1. The underlying raw data, in a nmon specific structured text file format, are available for further investigations.

More detailed and advanced information about the hardware and other system relevant information are shown in Figure 4.1, provided by a tool called inxi [160]. Inxi reports information about the used operating system, installed compilers, detailed insights about the used memory and the available CPU flags on the used topology. In addition to the metadata collected by inxi, information about the used tuned profile and whether hyper-threading is enabled or not are gathered. These information are available as a simple text file and can be used for further subsequent examinations of the system.

Scaling Mode

The BOOTABLE benchmark tool suite does not only offer the opportunity to benchmark existing hardware with regard to bioinformatics workloads, it also offers the possibility to measure the behavior of applications regardless of whether they come from the area of bioinformatics or not. Especially applications capable of multithreading can make use of the implemented scaling mode. The scaling mode automatically starts a benchmark run by using one, a quarter, a half and

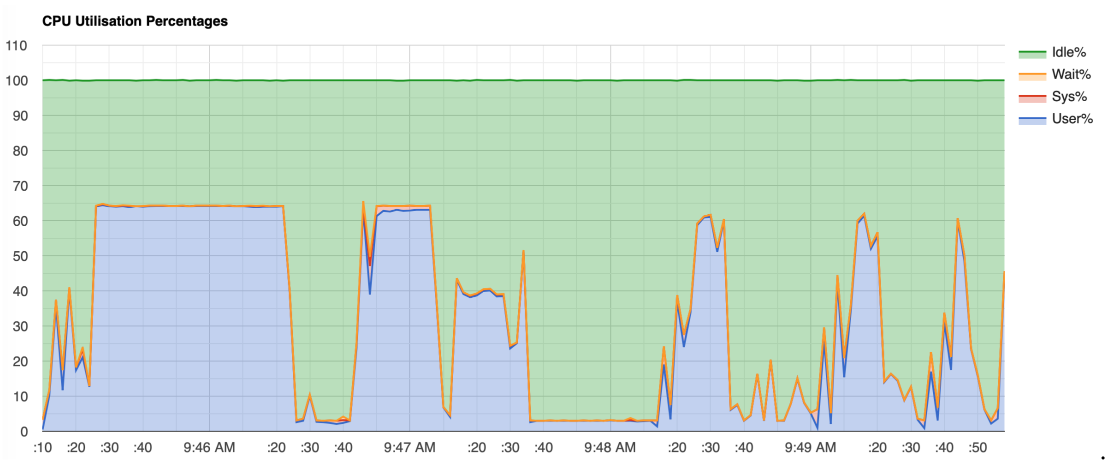


Figure 4.1: The raw data snapshots captured by nmon are converted into files in HTML format and are used to visualize the recorded data. This example shows different percentage values of the CPU utilization categories of the SPAdes assembly tool over time. Other available visualized resource categories are for example the CPU usage of every single core or disc read and write operations.

all of the available CPU cores of a system. All other parameters except the dataset can be chosen as desired. The dataset parameter is replaced by the scaling parameter. The most valuable output of the scaling mode are the plots generated specifically for this mode. An example scaling plot is shown in Figure 4.2.

Generic Tool Integration

All features mentioned above describe a solid benchmark environment. Especially the scaling mode and the collected metadata can be helpful in the case of application development. But of course not every tool desired by a user is already implemented, especially if it is under development. Therefore, a generic tool wrapper has been implemented. Once again the focus was placed on the user-friendliness. Therefore, a simple tool configuration file has been designed. A content example for bowtie2 is illustrated below.

```
Toolname:bowtie2_build
Dataset:Medium
Command:bowtie2-build --threads $cores --seed 42 dataset output
```

The `Toolname` parameter and the `Dataset` parameter can be freely chosen. They just support the users' convenience. The important parameter is the `Command` parameter. This line needs to contain the correct command line input with all necessary parameters to run the desired application. If a user wants to use the already built-in parameters for the number of threads or the scaling mode, it is necessary to set the variable `$cores` for the number of threads that

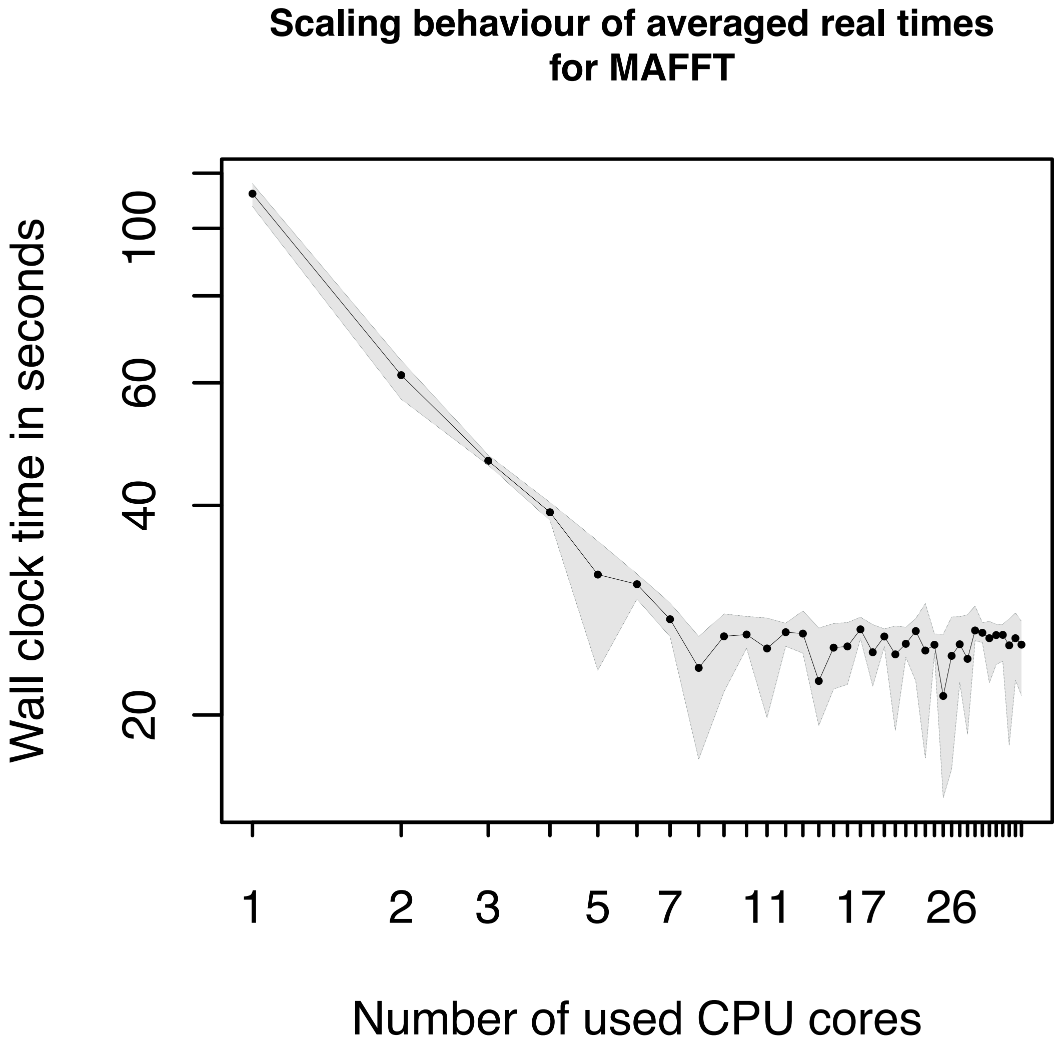


Figure 4.2: This example plot shows the scaling behavior of MAFFT for the long dataset. Both axes are on logarithmic scale in order to visualize whether a doubled amount of CPU cores halves the wall clock time and the chosen tool scales perfectly over time (y-axis) with the number of cores (x-axis) or not. The borders of the gray area show the maximal and minimal wall-time values measured. With these kind of plots it is easy and fast for a user or the developer to find out to what extent an application is able to scale. The minimal and maximal values measured, in case of multiple conducted runs (replicates), are illustrated by the boarders of the gray area.

should be used. Also, the built-in datasets can be used with one of the following variables: `$reference`, `$dataset`, `$dataset_idba`, `$dataset_clustalOmega`. If the generic tool option is set, BOOTABLE checks for a configuration file. If found, the file is parsed and the command is executed like any other command from an already integrated tool. The presented tool integration is purposely kept as simple as possible. It has been decided to use a user-friendly method with a simple and readable text, instead of a more advanced approach using XML, JSON, or CWL (Common Workflow Language) [161]. The generic tool integration feature enables BOOTABLE to overcome the lock-in to the domain of bioinformatics. Every desired tool, whether bioinformatics or not, can be used with the benchmark tool.

4.3 Tools and Datasets

The bioinformatics tools, applications and packages, used for BOOTABLE to generate a workload as close as possible to real word examples, are handpicked and based on the usage behavior of cloud users in the de.NBI cloud [5]. The applications are not only taken from various areas in bioinformatics, they also cover broader areas from life sciences like molecular dynamics and machine learning.

The same holds for the datasets, which are also carefully chosen. First, it has to be assured that the datasets are compatible with the tool they are used for. Second, the size of a dataset has a direct impact on the runtime, a user has the opportunity to choose between short, medium and rather long workloads regarding the computing time. The selection process is explained in detail by the subsections 4.3.1 and 4.3.7, already with regard to the subsequent scaling study.

4.3.1 Tool Selection

The tools were selected based on different criteria. First of all, a tool has to make use of multithreading in parts of their implementation, to get information about their scaling capabilities. Further, a tool should be well documented in terms of its execution, implementation and used algorithms. In addition, the selected tools should have as less dependencies as possible, like required tools or datasets. Of course it should be relatively popular and free for academic use, so that many users are able to profit from the findings of a scaling study. Another criteria of importance was the CPU core usage behavior. For example, one tool should use multiple cores throughout most of the execution time, whereas another one should use a single CPU core for longer periods and multiple cores only for a short amount of time, allowing broader tests in the process of a hardware procurement. This also includes a variation in terms of the algorithms and implementations used by the selected tools. Based on these criteria and communication to users of the de.NBI Cloud, the tools explained in the following subsections have been selected. Unfortunately, other popular tools could not be considered because they did not meet certain criteria. For example, the multiple sequence alignment tool MUSCLE [162] does not support multithreading. T-COFFEE [69] could not be considered either, since access to databases is necessary. This selection makes no claim to completeness regarding the selected tools. However, the selected tools cover a large range of application areas and users.

In the category of sequence assembly tools, Velvet, IDBA and SPAdes were selected, as all of them fulfill the criteria of multithreading, are well documented and straightforward to install. In addition, all of them use the general concept of de Bruijn graphs but in different variations, so it might be of interest to compare them with each other. Velvet was chosen because of the generated workload, as a large part of the implementation is single threaded but some parts are multithreaded. IDBA is selected because no article was available examining the scaling performance, the same applies for SPAdes. Further, SPAdes was chosen because it uses larger amounts of RAM, which is interesting to observe effects of larger datasets and RAM usage on the scaling behavior.

The category of sequence alignment is covered by the tools BBMap, Bowtie2 and BWA. All are well-known, especially Bowtie2 and BWA. Both use similar approaches for their index structures and for the alignment step. This makes it interesting to compare them with each other and with BBMap, which uses a different approach.

The tools Clustal Omega, MAFFT and SINA were chosen as representatives for multiple sequence alignment tools. Especially Clustal Omega and MAFFT have a large user base. The implemented approach of both tools is similar, therefore a comparison might be interesting. With SINA, a tool specialized on rRNA sequences, is selected to cover this specific topic.

For the topic of molecular dynamics, GROMACS has been chosen. GROMACS is widely used, freely available and very well documented. Of course there are other tools available, like CHARMM (Chemistry at HARvard Macromolecular Mechanics) [163] or Amber [164], which also have a large community and would have also been a valid choice.

For the area of AI, specifically machine learning, the TensorFlow framework and the CIFAR-10 application have been chosen, as TensorFlow is one of the mostly used frameworks and the CIFAR-10 image recognition application can be applied in the context of medical images. Another widely used frameworks that has to be named in this context is PyTorch [165].

A brief description of the tools and their application area is shown in the following, just as listed in Table 4.3 and a summary of the used algorithms in Table 4.4.

Application	Category	Version	Multithreading API	GCC
BBMap	Sequence Alignment	38.87	-	-
Bowtie2	Sequence Alignment	2.4.2	Threading Building Blocks	4.8.5
BWA	Sequence Alignment	0.7.17	Pthreads	4.8.5
Velvet	Sequence Assembly	1.2.10	OpenMP	7.3.0
IDBA-UD	Sequence Assembly	1.0.9	OpenMP/Pthreads	4.8.5
SPAdes	Sequence Assembly	3.12.0	Pthreads	-
Clustal Omega	Multiple Sequence Alignment	1.2.4	OpenMP	4.8.5
MAFFT	Multiple Sequence Alignment	7.475	Pthreads	4.8.5
SINA	Multiple Sequence Alignment	1.7.2	Threading Building Blocks	-
GROMACS	Molecular Dynamics	2018.3	OpenMP	7.3.0
CIFAR-10 model build	Machine Learning	1.4.0	Python Threadpool	-

Table 4.3: BOOTABLE benchmark applications, including their general category, version, used multithreading API and the GCC compiler version, whereas an already compiled binary is used for SPAdes BBMap and SINA. TensorFlow is installed via Python pip. For BBMap, implemented in the programming language Java, no note about the used multithreading API could be found.

Application	Algorithm
BBMap	Index: Sliding window with k-mers, Align: k-mer look up and scoring
Bowtie2	Index: BWT and FM-Index, Align: Seed and extend using Dynamic Programming
BWA	Index: BWT and FM-Index[a], Align: Seed and extend using Dynamic Programming
Velvet	De Bruijn Graph based on k-mers
IDBA-UD	Iterative de Bruijn Graph based on increasing k-mers
SPAdes	Multi-sized de Bruijn Graph, k-mers for graph construction only, graph theoretical operations
Clustal Omega	Pairwise distance matrix, guide tree (k-means), profile alignments (Hidden-Markov Models)
MAFFT	Pairwise distance matrix, guide tree (UPGMA), group-to-group alignment (iterative)
SINA	Index: Reference sequence selection used to create partial order MSA, Align: modified Needleman-Wunsch
GROMACS	Integration of Newton's equation of motions, force field (bonded and non-bonded interactions, domain decomposition)
CIFAR-10 model build	Convolutional neural network, multi-layer architecture, alternating convolutions, softmax classifier

Table 4.4: List of used benchmark tools. The algorithm column briefly describes the used algorithms to allow a comparison among each other.

[a]No parallelization

4.3.2 Sequence Assembly

Sequence assembly is a method to reconstruct genomes from reads, generated by sequencers. Genome or sequence assembly tools take advantage of the fact that gathered reads overlap. In the first step, small reads are assembled to larger fragments, known as contigs. Furthermore, contigs can be linked to each other in order to build scaffolds. In the end, all parts can be merged to reconstruct the whole input sequence [166].

Velvet: Velvet [167] is a *de novo* assembler that is specialized on sequencing technologies producing short reads. The input data (reads) are handled by a data structure using De Bruijn Graphs. The underlying De Bruijn Graph is manipulated in order to apply it to sequence assembly problems. The elements handled in the graph itself are not the whole reads, but shorter parts of it, so-called k-mers. The algorithm of Velvet can be divided into four parts: 1) Hashing, all reads are hashed to the corresponding value of k that determines the length of the k-mers. In short, the hashing step rewrites each read as a set of original k-mers combined with overlaps to previously hashed reads, which is called the roadmap. The same is done for the overlapping with subsequent reads. 2) Graph construction, the nodes created in 1) are now connected by edges, which is done by tracing reads trough the graph using the roadmaps generated before. 3) Error correction, after the initial graph is created, it undergoes a simplification step and an error correction. Velvet handles three different kinds of errors. Two topological ones named bulges and tips and a non-topological one called chimeric or erroneous connections. 4) Resolving repeats (Breadcrumb), repetitive regions can lead to problems by connecting and extending contigs. Velvet uses an approach of generating so-called long-nodes in combination with distance information of the paired end reads to span a repetitive region. At the end of the algorithm, most of the reads are mapped to the assembled contigs.

Regarding the time and memory complexity of Velvet, the main bottleneck is the graph construction step. The construction step, also as the whole algorithm depends on the input size, the number of reads and the chosen k, as it determines the number of nodes in the graph. A further non-negligible element is the complexity of the input data. The more complex it is the more tangled is the graph and the longer is the runtime especially for the error correction step. In order to point out the complexity of the error correction, Zerbino et al. explain it for the Tour Bus algorithm used for the bulge removal. The bulge removal strongly depends on the number of nodes in the graph as it is further indicated that the runtime of the search for bulges is based on the Dijkstra algorithm with a complexity of $\mathcal{O}(N \log_N)$, if implemented with a Fibonacci Heap. The number of nodes N further depends on the read coverage, error rate and number of repeats.

IDBA-UD: IDBA-UD is an iterative *de novo* assembler for single-cell and metagenomic sequencing data with highly uneven depth [168]. IDBA-UD uses a similar approach to Velvet by the use of a De Bruijn Graph data structure. The difference is, that IDBA-UD follows an iterative approach in building the graph and resolving errors. The first step of the algorithm is to construct a De Bruijn Graph from the input data (paired-end reads) with the lowest k, that is responsible for the length of the k-mers. After the construction, simplifications are done and also bulges and tips are removed. The second step targets the uneven depth problem of a sample through an approach using a relative depth, also taking the coverage of neighbored contigs into account. This approach lowers the false-positive rate of erroneously discarded contigs due to low coverage thresholds. The third step implements a read error correction on base level, based on alignments between reads and contigs with a given similarity reliability. The last step in the iterative part is a local assembly. This step assembles local reads to the already existing contigs in order to find k-mers that can fill gaps in the graph, even if they do not exist at all in the sample. After that, k will be increased and the algorithm starts again as long as the maximal k is reached. In the final step a scaffolding graph is build from the created contigs using the scaffolding algorithm of Li et al. [169].

Regarding the runtime and the complexity of the IDBA-UD algorithm similarities with Velvet, in terms of bottlenecks and input sizes can be found. Also for IDBA-UD applies, the larger the number of reads, the larger the De Bruijn Graph and the longer the runtime. Especially the number of nodes determines the memory requirements. As IDBA-UD uses an iterative approach, it uses a range of k to determine the length of the k-mers. Further time consuming steps are the computation of the progressive relative depths and also the local assembly. Again, both steps are dependent on the size of the graph and therefore on the number of reads in the sample, but also on their coverage and error rate.

SPAdes: The St. Petersburg genome assembler (SPAdes) [170] is another *de novo* assembly algorithm implementation, focused on single-cell and multi-cells of bacterial data, but it also covers standard multi-cell assembly. The used data structure and methods for bulge and tip removals are based on the multi-sized De Bruijn Graph construct. SPAdes uses another approach than Velvet or IDBA-UD. The construct of k-mers is only used for the first step of the algorithm, namely the construction of the multi-sized De Bruijn Graph. All subsequent operations are universal graph theoretical operations, independent of the k-mers. After the multi-sized De Bruijn Graph is constructed from the input data (paired-end reads) and their k-mers including the standard simplifications and error corrections (bulges, tips, chimerics), a so-called k-bimer adjustment is conducted. This second step of the SPAdes algorithm uses a number of different transformations in order to make use of the distance information through the paired-end reads (bireads). The next step creates a so-called paired assembly graph which is inspired by the concept of Paired De Bruijn Graphs [171]. The advantage of this kind of data structure is, that the contig sizes increase, compared to standard De Bruijn Graphs. In the last step, the paired assembly graph is used to construct contigs by simplifications to get the condensed edge representation paired assembly graph. At the end each read is mapped back to the contigs.

Bankevich et al. also measured the execution time regarding the already error corrected ECOLI-SC read set. The output of this measurement reveals that the first and the second step of the algorithm are the most time consuming ones. Especially the first one, the construction of the multi-sized De Bruijn Graph, takes the most time as it iterates over a list of different ks. Each iteration takes approximately the same amount of time. As noted by the authors, if the computations would be done for each possible k, it would slow down the algorithm and make it nonfunctional. The contribution of steps three and four to the overall runtime is rather small. Also to SPAdes applies, the larger the input data (number of reads) and therefore the number of nodes in the graph, the longer the runtime. Further, the number of chosen ks affects the runtime as it can lead to more tangled graphs and the error corrections of a tangled graph can get more time consuming. To reduce the runtime of the error correction steps and the complexity of the graph itself, Bankevich et al. propose to use BayesHammer [172] for a pre-error correction step.

4.3.3 Sequence Alignment

Sequence alignment can be defined as a procedure where a reference sequence is used and other sequences are aligned to the reference sequence. This method makes it possible to find similarities in specific regions of the reference sequence, that can be used to find further functional or evolutionary correlations [173].

BBMap: BBMap (Bestus Bioinformaticus Map) [174] is a splice aware alignment tool, designed for short and long reads resulting from DNA and RNA sequencing. To use BBMap, aligning reads to a reference sequence, an index of the reference sequence has to be created first. The algorithm used to create the index structure is based on a sliding window approach using k-mers. Every k-mer of a reference sequence is encoded as a sequence of two bit pairs for every base and transformed to integer values (keys) afterwards. The generated keys are used to construct the sizes array (index array) and the sites array, holding the required information for the mapping step, decreasing the lookup time. The first part of the mapping algorithm generates keys from the reads, similar to the index structure, but using an offset value as starting point for a k-mer. In the second step, a look up of the generated keys using the index structure is performed to find key hits. Found key hits are stored in a triplet structure holding the information where it is located, related to the reference sequence. These triplets are used in the third step to find possible alignments. The found alignments are evaluated by a scoring function and added to the resulting alignment. The algorithm stops if no triplets are left, which means that all keys have been processed [175]. All steps are also done for the reverse complement of the reference sequence and the reads to be aligned.

The space and time complexity depends on the number of bases of the reference sequence and also on the chosen k-mer size. Important for the alignment step is the number of reads and their number of bases, as well as the k-mer size. The shorter the k-mers the more sensitive is the mapping but the more amount of time and space is required. Larger k-mers would decrease the computational amount of time and space required, but would influence the sensitivity negatively.

Bowtie2: Bowtie2 [176] is the second version of the popular read alignment tool Bowtie, named after the implementation of the Burrows-Wheeler transform [177]. Both Bowtie versions make use of the Full-text Minute-size index (FM-Index) [178] to align huge amounts of reads in a short time. With the release of version two, Bowtie is extended by the ability to handle gapped alignments due to sequencing errors, true insertions or deletions. Before the first of the four steps in the alignment process of Bowtie2, the FM-Index of the reference genome needs to be created. After this is done, the first step of the alignment procedure starts by extracting so-called seeds from the input reads and their reverse complement. The seed length can be customized just as the interval the seeds are placed after each other. In the second step, an ungapped alignment is performed, aligning created seeds to the FM-Index. The output of this step is a set of Burrows-Wheeler ranges per seed string, these are called seed hit ranges. The third step uses the created seed hit ranges and prioritizes them depending on their range: the smaller the range, the higher the priority. The offset of the resulted rows are resolved and mapped back to the reference genome. In the end, candidates are created which looks worth to be extended through a dynamic programming alignment approach in step four. The applied alignment in step four allows Bowtie2 to construct a Sequence Alignment Map (SAM) including gaps.

The most time consuming step of Bowtie2 is the creation of the index structure, but the benefit of it is that it can be reused for other alignments and does not need to be computed again. Further, the generated index space is efficient as it takes up only 1.65 times the space of the original reference data and is therefore suitable for downloads from a general repository, as Langmead et al. suggest. The required time for the alignment step is short due to the advantageous data structure created in beforehand. Of course, both steps are dependent on the input size. The more bases the reference genome contains, the more time is required to build the index. For the alignment step the situation is similar, the more reads that need to be aligned, the longer lasts the alignment. The dynamic programming alignment in the last step can be accelerated by using multiple compute threads to solve smaller parts of the alignment in parallel. Otherwise, every alignment would require a time of $\mathcal{O}(NM)$ where N is the length of the reference sequence and M the length of the sequence to be aligned.

BWA: BWA (Burrows Wheeler Aligner) is a software package consisting of different alignment algorithms [177]. This work focuses on the BWA-MEM (Burrows Wheeler Aligner - Maximal Exact Matches) algorithm as it is the preferred one when using BWA [179]. As Bowtie2, BWA uses the FM-Index for the index structure of the reference sequence. In contrast to the other two available algorithms within the BWA package, BWA-MEM is the latest one. The core algorithm stays the same, using the seed and extend paradigm, but different improvements have been implemented. The initial seed alignment is done by searching for SMEMs (Supermaximal Exact Matches) [180]. In order to avoid mismappings resulting from missing seeds, a re-seed is performed. In addition, seeds are put together to chains greedily if they are colinear and close to each other. The generated chains are filtered to exclude short chains. Further, seeds are ranked by the length of the chain they belong to and afterwards by their seed length. A seed is dropped from the rank list if it is already contained in a previous alignment or extended with a banded affine-gap-penalty dynamic programming approach. The use of the banded dynamic programming approach leads to a linear time complexity in length of the query sequences. The general time and space complexity can be found in the original publication [177].

4.3.4 Multiple Sequence Alignment (MSA)

A multiple sequence alignment deals with a similar problem like a pairwise sequence alignment but instead of comparing only two sequences with each other, multiple sequences (three or more) are compared. The sequences to be compared can consist of nucleotides or amino acids. Results of such alignments can be used to find homologies between samples for subsequent phylogenetic analysis. [181].

Clustal Omega: Clustal Omega [182, 183] is a multiple sequence alignment tool for large numbers of nucleotides (DNA, RNA) and amino acids. Clustal Omega uses a guide-tree approach to determine the order of the subsequent alignments. The guide-tree is constructed from the pairwise distances between the input sequences in the first step of the alignment process, using the mBed algorithm. In order to speed up the distance calculations required for the tree construction, the k-tuple algorithm is used from prior versions of the Clustal MSA tool family. After the calculation of the pairwise distances, they are clustered using the bisecting k-means algorithm [184]. The result is a dendrogram, but it can only be used to guide the alignment process and not to reveal any phylogenetic relations between the sequences. The second step of Clustal Omega aligns sequences to larger groups, according to the branching order on the guide-tree, generating so-called profiles. The alignment is done by aligning two alignments with each other, at the beginning single sequences, later profiles with progressing tree traversal. The profile-profile alignment step makes an extensive use of HHalign [185], which is completely based on Hidden-Markov Models (HMMs). In order to get more accurate alignments using HHalign, every sequence and intermediary profile is converted into HMMs, which are aligned afterwards. After a full traversal of the guide-tree the alignment is finished.

Regarding the complexity and memory usage, Clustal Omega uses different methods to reduce the runtime and required space. A naive approach of the pairwise-distance calculations would require a time and space complexity of $\mathcal{O}(N^2)$ where N is the number of sequences. The usage of the mBed algorithm reduces the complexity to $\mathcal{O}(N(\log(N))^2)$ by randomly picking seed sequences up to $(\log N)^2$ of the available sequences and not of every sequence. Of course, the whole algorithm depends on the number of sequences, but also on their length. The number of sequences in first place affects the distance matrix calculations that would result in a required memory capacity of 14 MB for 10,000 sequences and 220 MB for 100,000. A full matrix instead would require 400 MB and 40 GB, that would make advanced computing hardware necessary.

The time complexity of the profile-profile alignment step can be represented as a function dependent on the number of sequences N, the lengths of the profiles L and the guide-tree shape. First, an MSA of N sequences needs $N-1$ profile-profile alignments, if the number of sequences is increased, the number of profile-profile alignments increases linearly and therefore the runtime increases also in a linear way. If the lengths of the sequences increase, the lengths of the profiles do so as well, because the construction of HMM matrices require a multiple of $L_1 \times L_2$, resulting in a quadratic increment in time. The shape of the guide tree contributes in a different way, if the tree is balanced the runtime is lower than using an imbalanced (chained) tree. The space complexity of an alignment of two profiles with length L can be described by the term $8 \times 6 \times L_1 \times L_2$, where 8 represents the space required for a double variable in Bytes and 6 the number of matrices, created by HHalign. For a single alignment of two profiles with a residue length of 100, a space of 480.000 Bytes of memory is required. Therefore, the limit of a system with 2 GB of memory is two profiles with a length of 6,688 positions each. In order to speed up the computation time, Clustal Omega uses the OpenMP library to enable multithreading for the pairwise distance calculations and the alignment match states computations [186].

MAFFT: MAFFT is a multiple sequence alignment program using the fast Fourier transform (FFT) method. The software was initially published in

2002 [187] and has been continuously developed since then. In a first step, MAFFT computes a distance matrix resulting from all-to-all pairwise comparisons of unaligned input sequences. The distance matrix is used to construct a guide tree, using the UPGMA method. The guide tree is used for the creation of an initial MSA by a group-to-group alignment at each node of the guide tree. This first, rough MSA is used as a new starting point for the second step, where again a distance matrix is calculated, a guide tree is constructed and used for the second, more precise MSA. In the last step, the initial alignment from step two is refined in an iterative procedure. The iterative procedure starts with the tree-dependent partitioning of an available MSA into two groups. The resulting groups are realigned using an approximate group-to-group alignment algorithm. The new MSA is scored by an objective function and replaces the old one, if the calculated score is higher. The algorithm stops if no further improvements are made.

The time complexity of MAFFT using the progressive method is expressed by the length (L) and number (N) of the input sequences with the term $\mathcal{O}(N^2L)\mathcal{O}(NL^2)$. The first part of the term is related to the guide tree construction and the second part to the group-to-group alignment. The space complexity is specified with $\mathcal{O}(N^2) + \mathcal{O}(L^2) + \mathcal{O}(NL)$ [188, 189, 190, 191].

SINA: SILVA Incremental Aligner (SINA), developed by Glöckner et al. [192], is a multiple sequence alignment tool specialized on ribosomal RNA (rRNA), using the rRNA databases provided by the SILVA project [193]. The algorithm of SINA uses a mix of partial order alignment and k-mer searching. In the first part, the reference sequence is selected based on a k-mer sequence search using PT server [194]. Based on the results, a logarithmic transformation is applied. Afterwards, SINA iterates over the matches. The matches are filtered by a rule set, deciding, whether matches are considered for the alignment template or not. In the second step, a directed acyclic graph is created from the result of the first step. Nodes of the graph represent an evolutionary unique base. Edges between nodes are drawn if a base occurs consecutively in any of the reference sequences. The resulting graph represents a partial order MSA (PO-MSA) [195]. The graph is used as data structure for the alignment step, applying dynamic programming. In order to align a candidate with the alignment template (PO-MSA), a modified version of the Needleman-Wunsch algorithm is used. By default, matching bases are rewarded with a score of 2 and mismatches with -1. In addition to the match and mismatch scores, two different weighting factors are used. Furthermore, SINA comes along with routines for unaligned sequence tails (overlap alignment) and insertions.

Due to the sequence selection step, prior to the alignment step, time and space complexity are decoupled from the size of the reference MSA.

4.3.5 Molecular Dynamics

Molecular dynamics deals with the simulation of atom and molecule movements and therefore their physical behavior. One use case in bioinformatics, for example, is protein structure prediction.

GROMACS: GROMACS (Groningen Machine for Chemical Simulations) [196] is one of the most used open source application regarding molecular dynamics. GROMACS is primarily used to simulate the dynamical behavior of biomolecules based on the integration of Newton's equation of motions. The forces acting on the individual particles within a molecular system govern the time evolution of the whole system. The molecular interactions are typically described within a force field comprising of the bonded interactions such as bond length, bond angle, dihedrals and the non-bonded electrostatics and van der Waals interactions. Within a periodic molecular system potentially all particles are interacting with each other exerting forces on each other. In practice, the short-ranged interactions dominate the dynamics of a molecular system declining with the increasing particle-particle distance. GROMACS uses a domain decomposition approach, splitting the component of the non-bonded interactions into domains consisting of spatially close particles. Thereby, the computational task can be distributed efficiently over multiple ranks or offloaded to a GPU (Graphics Processing Unit).

Nevertheless, a neighbor list has to be maintained and updated every few steps to keep track of particles moving from one domain to another. If all pair-wise electrostatic and van der Waals interactions are considered, explicitly molecular dynamics simulations scale by $\mathcal{O}(N^2)$ where N is the number of particles. If a particle mesh Ewald approach is applied to treat the long-range electrostatic interactions in Fourier space, the computational costs can be reduced to $\mathcal{O}(N \log(N))$ [197]. Furthermore, GROMACS takes advantage of hardware-specific SIMD (Single Instruction Multiple Data) kernels to parallelize the computational costly calculations such as the non-bonded and bonded force calculation or the particle mesh Ewald calculation in Fourier Space and, in addition, uses the OpenMP library to enable multithreading inside the spatial domains. To take full advantage of the SIMD features, GROMACS needs to be configured and compiled for the target CPU type. Hereby GROMACS is one of the most efficient molecular dynamics application available to the scientific community. Achieving a good GROMACS performance on a given hardware setup is a strong indicator for its capability.

4.3.6 Machine Learning

Machine learning is not specific to bioinformatics but there are many use cases where machine learning methods and algorithms can reveal interesting results applied to biological data, like medical images from X-ray photographs. Machine learning in general describes algorithms and statistical models for computer systems to improve their performance, based on specific training sets and unknown data.

CIFAR-10: The TensorFlow [198] open source framework offers a possibility to express and execute machine learning algorithms on a variety of systems and different datasets. The TensorFlow framework is used to build the model of a convolutional neural network, based on the CIFAR-10 training model application [199] and the corresponding CIFAR-10 dataset for image recognition. The goal of this model is to classify RGB images with 32x32 pixels into ten categories (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The underlying model consists of a multilayer architecture using alternating convolutions and nonlinearities with further fully connected layers ending up in a softmax classifier. The TensorFlow documentation proposes the following general workflow of three steps to implement a model. First, the model input routine. For this specific model the images are randomly cropped to 24x24 pixels. Afterwards, distortions are added in order to scale up the data size. Applied distortions are flipping images from left to right, set random brightness and contrast values. The second step covers the implementation of the model prediction, called inference, implementing the layer structure and convolution functions. In the end, this is required to classify unknown data by a trained model. For this specific case the output are probability values for the ten classes presented about how likely an unknown image fits each of the classes calculated from the generated logits. The third step describes the used methods to train the model on a specific dataset. The usually used method to perform a N-way classification is a multinomial logistic regression, also called softmax regression. Furthermore, losses are applied to all learned variables for regulation purposes. Further model training is done by a standard gradient descent algorithm with an exponential decay of the learning rate over time.

The time complexity to train and test a computational neural network in general depends on six variables, the index of a convolutional layer, the number of such layers (depth), the number of filters (width) and their spatial size (length), the number of input channels, and the spatial size of the output feature map, where the convolutional layers are the most time consuming steps [200]. He and Sun tested different models changing one parameter and keep all others fixed. The outcome of their work is that more layers (deeper) and smaller filters are able to reduce the time complexity whereas depth and width do not show any significant priorities to each other. The same holds to the relation of width and filter sizes. In the end they also claim, that the depth influences the memory consumption and can have a huge impact due to the backward propagation.

4.3.7 Datasets

The benchmark datasets are hand picked and come along with BOOTABLE. The different datasets were chosen regarding their runtime, to provide the possibility of rather short benchmarking runs up to longer runs. The used datasets are real scientific datasets and not created artificially. A list of the provided datasets is shown in Table 4.5, followed-up by Table 4.6, listing the datasets grouped by their runtime categories (short, middle, long). In the following, the various datasets are presented in detail, grouped by their application area.

Reference: In order to use Bowtie2, BWA or BBMap to align a sequence, the index structure of the reference genome has to be build first. This step is computationally expensive and depends on the input size of the reference sequence. For Bowtie2 and BBMap the largest of the three datasets (category long) is the full human genome (*GRCh38_full_analysis_set_plus_decoy_hla.fa*), taken from the 1000 Genomes Project [201]. For the category middle and short the datasets *DRR001012.fastq*, *DRR001025.fastq* are used. These are taken from the sequence read archive [202], hosted by the European Bioinformatics Institute (EBI). For BWA the same datasets have been used but the index structure has been provided directly as the index creation of BWA is not multithreaded and would only contribute constantly to the scaling results. For Bowtie2_align the same dataset is used for the categories short and middle but against different references, generating different kinds of runtimes.

Assembly: The integrated assembly applications Velvet, IDBA-UD and SPAdes use four different datasets, all extracted from the 1000 Genomes Project. The largest one is the dataset *ERR2510006.filt.fastq* used by all three tools. Due to incompatibilities of the dataset *ERR016155.filt.fastq* with IDBA-UD, the datasets *ERR015528.filt.fastq*, *SRR741411.filt.fastq* are especially added for it. *ERR016155.filt.fastq* is used for both, the short and middle categorized runs as the experienced runtimes fit both categories.

MSA: The MSA dataset for Clustal Omega and MAFFT is taken from the NCBI hosted Genbank repository [203]. Even after longer research for the different runtime categories no perfectly matching datasets could be found. Therefore, the original dataset *wgs.ANCA.1.fsa_aa* was taken and the first 32, 71 and 96 sequences are used as input for the categories short, middle and long. Care was also taken to keep the file structure intact. For SINA different datasets are used, as SINA is specialized in rRNA sequences. As reference index the recommended dataset of the small subunit (SSU) Ref NR 99 in version 138.1, provided by the SILVA rRNA database project is used. As this is the only reference dataset that is used, the index structure was computed in beforehand as

Dataset	Size
ERR2510006.filt.fastq[a]	1600
ERR016155.filt.fastq[b]	284
ERR015528.filt.fastq[c]	226
SRR741411.filt.fastq[d]	136
DRR001012.fastq[e]	683
DRR001025.fastq[f]	2000
GRCh38_full_analysis_set_plus_decoy_hla.fa[g]	3100
wgs.ANCA.1.fsa_aa[h]	0.5
SILVA_138.1_SSURef_NR99_12_06_20_opt.arb[i]	541
GTDB_bac-arc_ssu_r86.fa[j]	31
RefSeq-RDP16S_v2_May2018.fa[k]	23
OE-38_R1.fastq[l]	3.7
CIFAR-10[m]	163
ADH_bench_systems (adh_cubic)[n]	59

Table 4.5: BOOTABLE benchmark datasets with sizes in megabytes

the contribution to the total scaling is constant. For the alignment benchmarks of SINA, two datasets (categories middle and long) have been taken from the DADA2 project[6]. The dataset belonging to the category short[7], has been taken from the Orchard project [204].

GROMACS: As dataset for GROMACS, also proposed by the developers, a part of the alcohol dehydrogenase enzyme embedded into water within a cubic box is used. In order to produce longer runtimes, the step numbers are increased starting from 10,000 (short), 30,000 (middle), up to 50,000 for a long benchmark run.

Machine Learning: For the CIFAR-10 model build application the CIFAR-10 dataset is used. The dataset is provided by the Canadian Institute For Advanced Research and is one of the standard image sets related to machine learning and image recognition. It consists of 60,000 colored images with a size of 32x32 pixels, divided into 10 classes, with 6,000 images per class. The CIFAR-10 dataset is used for all runtime categories. The runtime is controlled via the used step sizes of 1,000, 2,500 and 5,000.

[6] 10.5281/zenodo.2541239

[7] 10.5281/zenodo.803376

[a] `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/HG00110/`

[b] `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/HG00125/`

[c] `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/HG00106/`

[d] `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/HG00099/`

[e] `ftp://ftp.sra.ebi.ac.uk/vol1/fastq/DRR001/DRR001012/`

[f] `ftp://ftp.sra.ebi.ac.uk/vol1/fastq/DRR001/DRR001025/`

[g] `http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/`

[h] `ftp://ftp.ncbi.nlm.nih.gov/genbank/wgs/A/wgs.ANCA.1.fsa_aa.gz`

[i] `https://www.arb-silva.de/download/arb-files/`

[j] `https://zenodo.org/record/2541239/files/GTDB_bac-arc_ssu_r86.fa.gz`

Tool	Short	Middle	Long
BBMap	DRR001012.fa	DRR001025.fa	GRCh38
Bowtie2_build	DRR001012.fa	DRR001025.fa	GRCh38
Bowtie2_align	ERR016155.filt.fastq	ERR016155.filt.fastq	ERR251006.filt.fastq
BWA	DRR001012.fa	DRR001025.fa	GRCh38
Velvet	ERR016155.filt.fastq	ERR016155.filt.fastq	ERR251006.filt.fastq
IDBA-UD	SRR741411.filt.fa	ERR015528.filt.fa	ERR251006.filt.fastq
SPAdes	ERR016155.filt.fastq	ERR016155.filt.fastq	ERR251006.filt.fastq
Clustal Omega	wgs.ANCA.1_200.fsa	wgs.ANCA.1_400.fsa	wgs.ANCA.1_500.fsa
MAFFT	wgs.ANCA.1_200.fsa	wgs.ANCA.1_400.fsa	wgs.ANCA.1_500.fsa
SINA	OE-38_R1.fa	RefSeq-RDP16S_v2_May2018.fa	GTDB_bac-arc_ssu_r86.fa
GROMACS	10000	30000	50000
CIFAR-10 model build	1000	2500	5000

Table 4.6: Benchmark datasets assigned to the according tool and dataset parameters (Short, Middle, Long) of BOOTABLE.

4.4 Performance and Scaling Measurements

BOOTABLE has been initially developed to let resource providers explore and evaluate their hardware with regard to bioinformatics workloads. But with the rich set of features to measure the consumption of various resources it has become a valuable tool for application developers and users, too. In this part of the thesis the main focus is on the scaling mode. The original benchmark execution script has been modified to capture all possible numbers of threads starting from one up to the maximum, increasing by steps of one, in order to perform the scaling study and measure the performance of the integrated tools in different compute environments. The modified version is available in the corresponding GitHub repository[8]. The following performance and scaling measures were performed by BOOTABLE using five replicas for each CPU core run, all available predefined datasets (short, middle, long), and all available tools.

4.4.1 System Environment

In total, three different kind of computational environments have been used, but always on the same underlying hardware. First, every benchmark has been conducted directly on a server (bare-metal), meaning BOOTABLE has been used without any virtualization in between. Second BOOTABLE has been installed in a virtual machine (VM) embedded in an OpenStack cloud, that is part of the de.NBI Cloud [5]. And third, BOOTABLE has been installed using the Docker image inside of a VM. The OpenStack cloud software is used to launch

[k] https://zenodo.org/record/2541239/files/RefSeq-RDP16S_v2_May2018.fa.gz
[l] https://zenodo.org/record/803376/files/OE-38_R1.fastq.gz
[m] https://www.cs.toronto.edu/~kriz/cifar.html
[n] https://ftp.gromacs.org/pub/benchmarks/
[8] https://github.com/MaximilianHanussek/BOOTABLE/tree/master/mods

and manage the virtual machine but has no direct impact on the VM itself, every other virtualization management software should be just as good. Docker has been chosen as it is one of the most used container technologies especially in the community of bioinformatics. The hardware consists of a two socket server system equipped with Intel Xeon Gold 6140 ”Skylake” CPUs with 18 cores each, which adds up to 36 CPU cores in total operating at a base clock rate of 2.3 GHz. Hyper-Threading has been disabled and the ratio of physical CPU cores to virtual CPU cores was 1:1, which means every virtual core has been mapped to exactly one physical core. Further, the system hardware provides 1.5 TB of RAM and a local SSD with a capacity of 480 GB. For all three environments, CentOS in version 7 has been used as operating system. The virtual machine environment is using QEMU-KVM as virtualization software, whereas Docker has been used in version 1.13.1 and overlay2 as storage driver. The dockerized version of BOOTABLE has been created from a Dockerfile, that is similar to the BOOTABLE installation script. From the Dockerfile a Docker image has been build on the same server type as the benchmarks has been performed on. For the scaling benchmark, the image has been pulled from Docker Hub and run as an interactive container without any further adjustments.

4.5 Results

The performance measurements are divided into two parts. The first part covers the average wall-times of the specified tools, whereas the second part presents the scaling behavior results.

4.5.1 Performance Measurements of Virtual Environments

The performance of the different environments was evaluated measuring the wall-time of every tool from the start of the execution to its end. The wall-time has been also taken for the evaluation of the scaling performance. The choice of a measure should reflect the results as good as possible and as intuitive as possible. From a user’s point of view, the most likely question is: ”How long will it take to get the results?” The wall-time would be one of the most interesting and simple units. As the mass of numbers and results would be overwhelming, a selection of the relevant results and averaged values is presented. In the following the measurements of the bowtie2-build application are neglected as the creation of the index structure involves random starting points, which can lead to larger differences regarding the wall-times. To make the results more comparable across all datasets and environments, these measurements are ignored if not noted otherwise. Further, it has been experienced that not all CPU core numbers can be used with GROMACS, reasons for that are discussed in Section 4.6. These

Environment	Overhead [%]			
	1 core	2 cores	...	36 cores
BM	0	0	...	0
BM-VM	7.71	8.20	...	9.67
BM-Docker	12.15	20.43	...	23.75
BM - Docker on bare-metal	9.02	10.54	...	4.65
VM-Docker	4.25	11.89	...	13.08

Table 4.7: Example of summed up and averaged percentage values describing the overhead for the short dataset in different envrionments. Bare-metal (BM) which is the environment the virtual ones have to be compared to. The virtual machine (VM) environment, running BOOTABLE inside a VM. Docker, using the BOOTABLE Docker image inside a VM. Docker on bare-metal, BOOTABLE installed via the Docker image directly on the hypervisor without using a VM. The second named environment in the first column shows an overhead to the first named one.

failed runs are excluded from the performance measurements of the specific CPU configurations, as their calculated percentage values would lead to a bias in the overhead results.

First of interest was to determine the overhead of the different environments (bare-metal, VM, Docker), according to the different sized datasets (short, middle, long). To gain insights, the raw wall-time values were transformed into percentage values, a part of these values is shown in Table 4.7. The final overhead values for every computing environment and dataset are presented in Table 4.8 and were calculated as following. First, the relative overhead of every tool for a given dataset and core configuration between the bare-metal environment and one of the virtual computing environments (VM or Docker) is calculated from the raw wall-time values. Afterwards, the average percentage for every core configuration for all tools is calculated to get the configuration overhead stats per core. Finally, the average per core percentages are summed up and divided by the number of core configurations, excluding the not ran GROMACS benchmarks for special CPU configurations. This was done for all environments, combined with the different CPU core configurations.

The final results show that virtualization on VM level leads, on average, to an overhead of around seven to ten percent related to the bare-metal values as reference. Regarding Docker, the overhead is even higher than for the VM with 15 to 24 percent, compared to the bare-metal values. It can also be seen that the increase of the used datasets in size and complexity leads to nearly the same overhead results with deviations in the range of around three percents. If the raw time values are illustrated as a stacked bar plot (shown in Fig 4.3), more details regarding the used CPU numbers are revealed. One specific detail are the wall-time values of the bare-metal single core performance. The values are lower than the ones of the virtual environments. The single core performance of a VM,

Environment	Dataset	Overhead [%]
VM	short	7.73
Docker	short	22.54
Docker on bare-metal	short	6.63
VM	middle	10.51
Docker	middle	24.45
VM	long	7.12
Docker	long	14.94

Table 4.8: Average overheads over all used CPU core numbers of VM and Docker runs and one run of Docker directly on bare-metal. The overhead numbers are related to the bare-metal benchmarks, expressed in percentages, including the used dataset.

for example using the middle dataset, is around 13 percent lower, meaning the benchmarks inside a VM need 13 percent more time for the same amount of work compared to a bare-metal machine. But with the increasing number of cores, the overhead values of the VM environment decreases down to the average values. For the Docker environment inside a VM the overhead is nearly constant in the range of around 20 to 25 percent. A behavior with decreasing values has not been observed over all three dataset categories.

Taking a closer look at the differences between the results of the VM environment and Docker, some cases were found where the measured wall-time of a virtual environment was lower than for the bare-metal environment. It also occurred that the Docker environment required less time than runs on a virtual machine. These cases have been observed for all datasets but it occurs sporadically for different tools and different core configurations in the range of two percent with some rare outliers of ten to twelve percent, mostly related to Clustal Omega. Furthermore, a benchmark of the short dataset using Docker on bare-metal has been conducted for the sake of completeness. The results show that the difference of the wall-times between the bare-metal environment and Docker on a bare-metal environment can be compared to the one of the VM environment with a similar overhead value of around seven percent. Reasons why the Docker on bare-metal setup has not been evaluated with the larger datasets can be found in Section 4.6.

4.5.2 Scaling Behavior

In this section, the focus is shifted towards the scaling behavior of the different tools. In order to visualize the discovered results in tabular format, the scaling behavior of the tools are put into one of three categories, yes it scales, no it does not scale and partially, if the behavior can not be sorted into one of the categories yes or no. The results are illustrated in Table 4.9.

To get a direct impression whether a tool scales using additional CPU

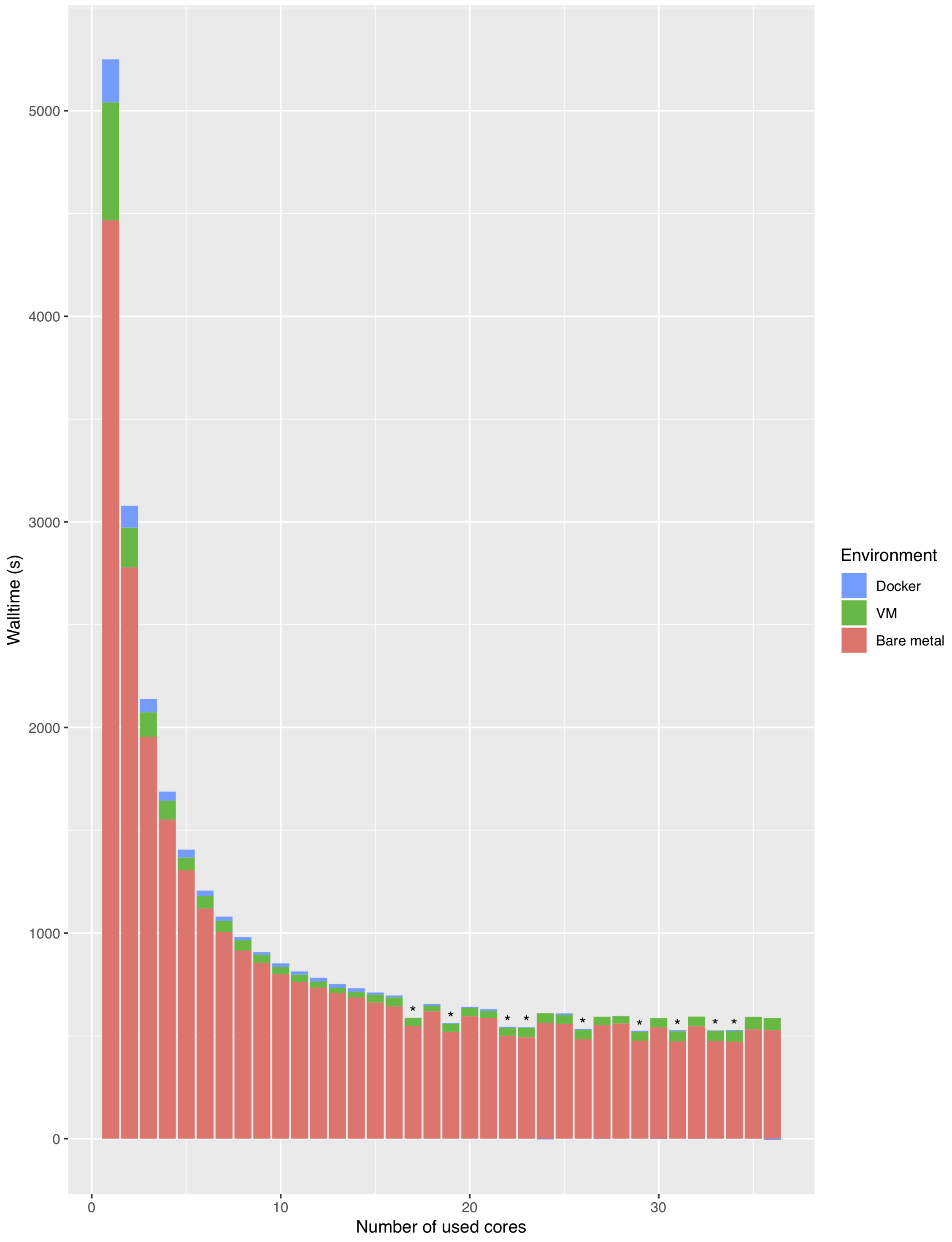


Figure 4.3: CPU core number-wise comparison of summed up wall-times for the different computing environments based on the long dataset, presented as stacked bar chart. The stacked parts need to be seen as an addition to the already existing part under it. Red bars represent the results of the bare-metal environment, green the additional time in a VM and blue using a Docker container. The bars marked with a star indicate CPU core numbers where GROMACS did not allow to start the simulation.

Application	Yes	No	Partially
BBMap	✓	-	-
Bowtie2_build	✓	-	-
Bowtie2_align	✓	-	-
BWA	✓	-	-
Velveth	-	×	-
Velvetg	-	×	-
IDBA	-	-	✓
SPAdes	✓	-	-
Clustal Omega	-	×	-
MAFFT			✓
SINA	✓	-	-
TensorFlow	✓	-	-
GROMACS	✓	-	-

Table 4.9: Classification of the used applications by the scaling results into one of the three different categories Yes, No and Partially, related to the long dataset. The long dataset also represents the datasets short and middle. Category Yes means that the application scales satisfactory with the number of used cores. Category No means it does not scale. Partially means that for some numbers of CPU cores the application scales satisfactory, for others not.

cores, scaling plots with logarithmic x and y axes have been created. In the following, the scaling plots of the long dataset in a bare-metal environment are presented. The scaling plots of the different environments do not highly differ from each other. The same applies to the different datasets, they only differ in the scale of the measured wall-times. A first look on the scaling plots of the sequence alignment tools, presented by Fig 4.4, shows a quite diffuse behavior for bowtie2_build (top left). Especially for higher numbers of CPU cores the values are alternating. An explanation of this behavior is given in Section 4.6. For bowtie2_align (top right) a nearly perfect linear graph is shown, which points towards a linear behavior for the alignment process. For the short and middle datasets, the scaling is not as linear as for the long one but still satisfactory. A linear behavior is mostly shown for BWA using the MEM algorithm (bottom right), except for the last core configuration using 36 cores, showing an unusual wall-time increase. The linear behavior from the alignment tools before applies also to the third alignment tool, BBMap (bottom left). The scaling is close to linear but with less improvements towards the maximal number of available cores for the short dataset. In general, all selected tools of the sequence alignment category showed a sufficient scaling performance.

The results of the *de novo* assembly tools are shown in Fig 4.5. The *de novo* assembler Velvet is split into two execution steps and considered as two separate tools (Velveth, Velvetg). Upon first look Velveth shows a quite diffuse behavior with values going up and down, but the range of the lowest and the highest measured wall-time value is in a range of 1 second and therefore nearly constant, according to the used CPU cores. Velvetg shows a similar behavior as Velveth

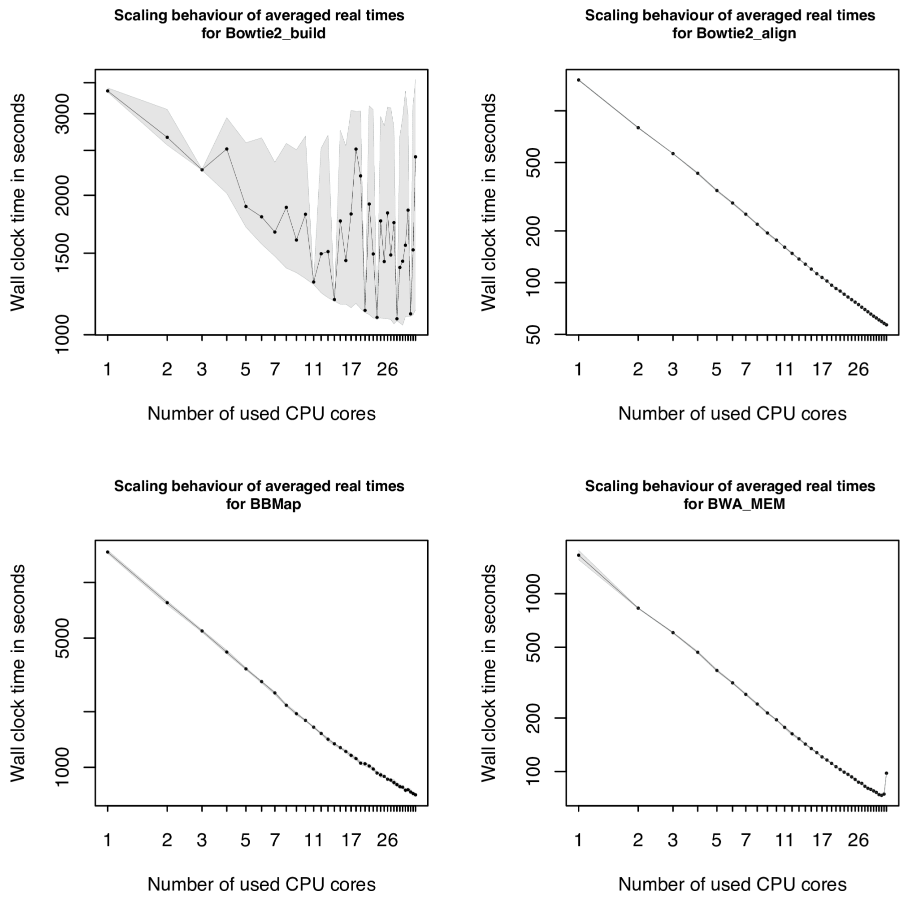


Figure 4.4: Graph representing the scaling behavior of Bowtie2_build (top left), Bowtie2_align (top right), BBMap (bottom left) and BWA (bottom right), using average values from five replicas from the long dataset. Both axes are scaled logarithmically to visualize whether the benchmarked tools benefit from the additional CPU cores or not. The values on the x-axes represent the number of used cores, whereas the y-axes show the measured wall-time in seconds. Please note that the y-axes of the different scaling plots cover different ranges of measured wall-times. In addition to the curve, the borders of the gray area show the minimal and maximal wall-time values measured.

but in a range of around ten percent between the minimum and maximum values. The bottom left plot of Fig 4.5, shows the scaling results of the assembly tool IDBA. In the beginning, the wall-times scale almost linearly regarding the used CPU cores, until 16 cores. Afterwards, the values stagnate and the graph flattens out. With increasing core numbers towards the maximal number of 36 cores the wall-time values even increase. The last tool of the assembly category is SPAdes. The scaling graph is shown in the bottom right. SPAdes has the highest runtime compared to the other assembly tools (Velvet, IDBA-UD). The graph shows an almost linear scaling behavior with a light bump for two cores and a light flattening starting from 17 cores on.

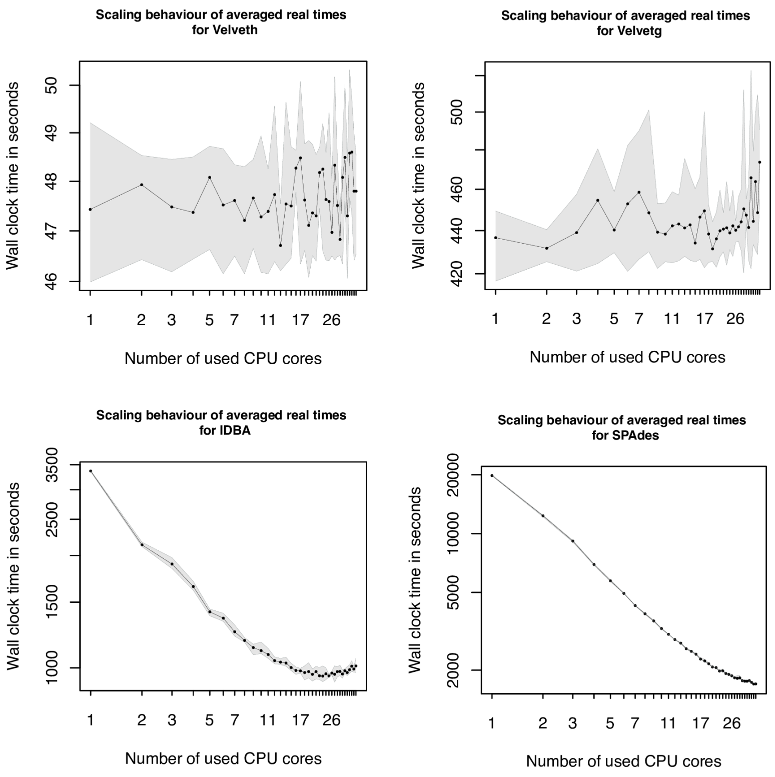


Figure 4.5: Graphs showing the scaling behavior of Velveth (top left), Velvetg (top right), IDBA-UD (bottom left) and SPAdes (bottom right) of the long dataset. The x-axes represent the number of used cores in the range of 1 to 36. The y-axes show averaged wall-time values in seconds in different ranges, depending on the tool. Both axes are scaled logarithmic to visualize whether the benchmarked tools can halve the wall clock time by using doubled numbers of CPU cores or not. In addition to the scaling graph, the minimal and maximal values of the five conducted runs are illustrated by the borders of the area in gray.

The scaling behaviors of the MSA tools are illustrated in Fig 4.6. The graph on the left shows the behavior of Clustal Omega. It reveals a contrary view on the scaling. The lowest wall-time values are achieved using only one core. The more CPU cores are used, the higher are the wall-time values. With the last configuration, the usage of 36 cores, the total wall-time value has almost doubled. MAFFT shows a different scaling behavior than Clustal Omega. It scales almost linear with smaller up and downs up to 8 cores. The tool SINA, specialized on rRNA, shows a quite linear scaling behavior (graph on the right in Fig 4.6), with slightly less decreasing wall-time values for higher numbers of CPU cores.

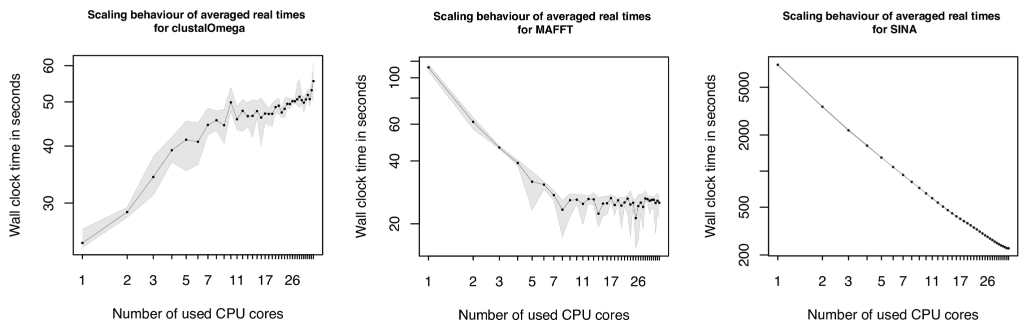


Figure 4.6: The graphs illustrate the scaling behavior of Clustal Omega (left), MAFFT (middle) and SINA (right) using the long dataset. The x-axes values are in the range of 1 to 36 and represent the number of used cores. The y-axes show averaged wall-time values in seconds. The ranges of the y-axes differ depending on the tool. Both axes are scaled logarithmically to visualize whether the benchmarked tools benefit from the additional CPU cores or not. To indicate the minimum and maximum wall-time values of the five executed runs, a gray area has been inserted, whose boundaries represent the lowest and highest measured values.

The fourth series of scaling graphs (see Fig 4.7) shows the scaling behavior of the CIFAR-10 TensorFlow application and GROMACS. The TensorFlow application scales practically linear with a light bend after ten cores and a very light flattening effect at the end. For the molecular dynamics simulation tool GROMACS, a rather odd behavior can be observed for the CPU core numbers 17, 19, 22, 23, 26, 29, 31, 33 and 34. For other numbers of CPU cores the graph shows a regular behavior, regarding the measured values from the lower CPU core numbers. The reason for this behavior is clarified in Section 4.6.

4.6 Discussion and Analysis

The Discussion and Analysis section is divided into two parts. Each part discusses the found results and gives explanations for the observed behaviors.

4.6.1 Virtual Environment Comparison

The obtained results show that the different virtualization environments have an effect on the measured wall-times. The bare-metal setup performs best, followed by the VM benchmarks and finally the Docker environment. This can

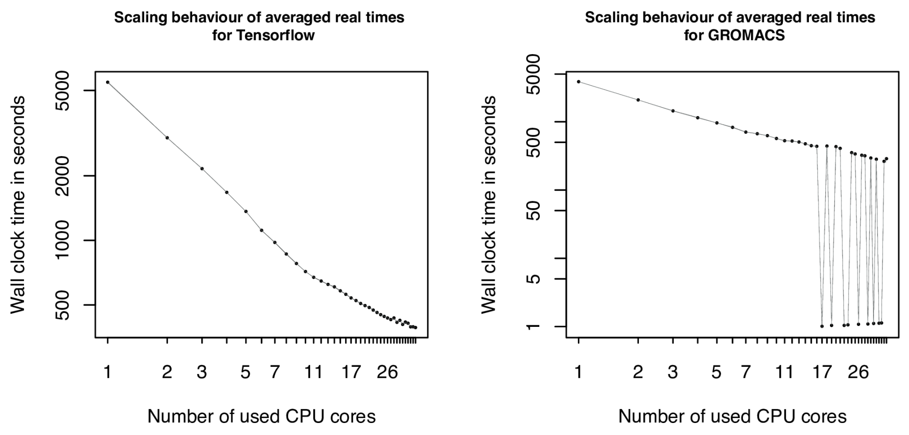


Figure 4.7: Scaling graphs showing the TensorFlow application CIFAR-10 (left) and GROMACS (right) using the long dataset. The number of used cores is represented on the x-axes, whereas the y-axes show the on average measured wall-times in different ranges, depending on the tool. The x- and y-axes are scaled logarithmically to visualize whether the benchmarked tools benefit from the additional CPU cores or not. The borders of the gray area illustrate the minimal and maximal measured values for each used core, but are barely visible because only tiny deviations from the different replicas have been measured.

be explained through the usage of the hypervisor layer (KVM). The required virtualization adds an additional layer of abstraction to the environment and therefore a loss of performance is observed. The overhead can be specified with around seven to ten percent over all datasets for the VM environment compared to the performance of the bare-metal setup. Using Docker, the performance of the benchmarks is worse compared to the VM setup and thus also worse than the bare-metal performance. For all datasets, an overhead of more than 15 percent compared to the bare-metal values have been identified. The overhead between the VM and Docker environments is rather high, with around 8 to 15 percent on top of the VM performance. The additional loss of performance can be explained through the additional virtualization layer as Docker containers were run on top of the hypervisor layer inside a VM. Besides the additional virtualization layer of the Docker engine, the used file system of the executed container might be responsible for a light performance loss. The recommended overlay2 driver was used, but still there exists some overhead. This also seems to be the reason for the larger overhead compared to the VM environment. A good example that shows the additional overhead created by the additional layers, is the CIFAR-10 application using the TensorFlow framework. TensorFlow is mostly written in C and therefore makes lots of system calls. Due to the high number of system calls, the computations executed through the TensorFlow framework also spend a larger amount of time for these system calls. Executed in a bare metal environment, the system calls are faster than on a VM or inside a container as there is no need to translate the system calls from the guest operating system to the host and back. Based on the measured wall-time values of the CIFAR-10 tool, this can result in an overhead of 25 to 30 percent. Further, supporting evidences for this assumption are provided by GROMACS, that shows a similar system call behavior like the CIFAR-10 tool and similar overhead results regarding the different execution environments.

The additional benchmark run of the dockerized version of BOOTABLE on bare-metal showed that the wall-times are close to the measured values of the VM environment. The performance of Docker on a bare-metal instance was not evaluated beyond the short dataset as there are hardly any major infrastructure operators that allow such a setup due to security reasons. But for smaller infrastructures it might be of interest, so this scenario has been taken into account.

Furthermore, the question arose whether the size of the datasets can have an influence on the scaling behavior. Of course, the larger and more complex datasets showed an increased wall-time but apart from that, only some odd behaviors regarding the scaling behavior of Clustal Omega were recognized, which is discussed in Section 4.6.2.

4.6.2 Scaling Behavior

After the investigation of the various scaling plots, no effects on the scaling behavior triggered by the different execution environments were observed. The same applies to the different datasets, which also have no or only a very small effect on the scaling performance. The only difference is observed for IDBA-UD which produces a flat graph for higher CPU core numbers on the long dataset instead of a rising graph for the short and middle dataset. This behavior can be observed for all execution environments.

The scaling performance of the Bowtie2 build tool does not look like it would scale, but taking a closer look on the raw wall-time data reveals some fluctuation. The values used for the generation of the scaling plots are the mean of five benchmark runs for every CPU core configuration. Some of the five replicas showed clearly shorter wall-times, whereas others showed higher ones. The reason for this is that Bowtie2 chooses a random starting point based on the random number generator seed. Depending on the generated random number, the runtime can increase by factors of 2 or 2.5. This of course affects the calculated average values. If only the lowest values would be taken ignoring the outliers, the graph would show an almost linear scaling behavior. Instead, the Bowtie2 aligner shows a nearly perfect linear scaling behavior, with each CPU core doubling almost halving the measured wall-times. This is also true for the other tools of the sequence alignment tools category, BBMap and BWA. For BWA, an increase of the wall-time using 36 cores has been seen, which might be related to operating system processes that also require some CPU time. In general, these results coincides with the knowledge that the alignment of two sequences is a well-known and studied problem in bioinformatics, that can efficiently be solved using a dynamic programming approach. This approach is applicable because the problem can be divided into smaller subproblems (partial alignments), which can be processed independently by each other. This makes it possible to distribute the load efficiently on the available CPU cores.

A more complex problem in bioinformatics is the alignment of multiple sequences covered by the applications Clustal Omega, MAFFT and SINA. Clustal Omega provides a multithreading option, but during the scaling study the best performance was observed using a single core. Furthermore, the provided graph in Fig 4.6 shows almost continuously increasing wall-time values for higher numbers of used CPU cores. The results are also not comparable to the one of Velvet where the usage of multiple compute cores do not lead to a strong speed up. In the case of Clustal Omega, it leads to a doubling of the runtime, comparing the values of the best performance for one core to the worst using 35 or 36 cores. Also, there is no sweet spot where the wall-time would first decrease and then increase after a specific number of cores. If one switched to the verbose mode of Clustal Omega, the output reveals that the longest part is the progressive alignment step, which seems to be affected negatively by the usage of multiple cores if datasets with a small number of sequences are used. It can be seen from the documentation of Clustal Omega that the procedures to calculate the pairwise distances and parts of the HMM building procedures are parallelized. It has been approved that these parts benefit from the additional, cores but the subsequent progressive alignment step negates the achieved benefit. To verify the assumption that the small sizes of the datasets are responsible for the resulting overhead, an additional test on the full ANCA.1 dataset was performed. The first information gathered from the provided output is that the mBed algorithm is only used for more than 100 sequences to calculate the pairwise distances. The second finding was that a test run on the full dataset (1258 sequences) using a single CPU core took two times longer than the same dataset using the maximum of 36 cores. But using only 19 cores led to an even better performance, which seems to be the sweet spot in this case. Using more CPU cores did not lead to a further reduction of the measured wall-time, on the contrary, the numbers slightly increased. The full scaling plot is shown in Fig 4.8. The second representative of the MSA category is MAFFT. The observed stagnation of the scaling behavior with the usage of more than 8 cores could be caused on the one hand by the used algorithm itself or on the other hand by the implementation. Both could lead to a limitation of the scalability to 8 CPU cores. In contrast to Clustal Omega and MAFFT, the third MSA tool, SINA, scaled very well. The reason for this might be the different setup and the usage of a reference index, that would put SINA in between the sequence alignment category and the multiple sequence alignment category.

For two of the three *de novo* assembly tools a rather poor to medium scaling performance has been observed. Although Velvet offers multithreading, most of the computations are not parallelized and therefore most of the workload is done

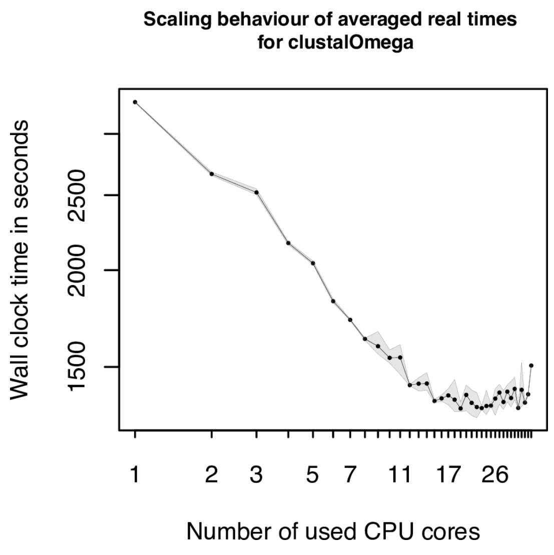


Figure 4.8: Additional scaling plot showing the scaling behavior of Clustal Omega, using the full ANCA.1 dataset. The x-axis represents the number of used cores where the y-axis illustrates the measured wall-time in seconds. Due to the logarithmic x- and y-axes, the scaling performance is made visible. The borders of the gray area show the measured minimal and maximal wall-times from the multiple replicas.

using a single core. With that knowledge, it is not surprising that the scaling plots show some diffuse fluctuation, but in a rather small range. Therefore, it does not make a large difference if more than a single core is used. The second *de novo* assembly tool is IDBA-UD. It also offers multithreading and the performance measurements show that it makes use of the parallelization in most parts of the algorithm. The determined change in the scaling behavior, if more than 16 CPU cores are used, let assume that the implemented parallelization procedures are somehow limited to this number or that is it not possible to improve the parallelization because of the algorithm. To the best of our knowledge, there seems to be no reason why the algorithm should not make use of more than 16 compute cores, so the limiting factor might be the implemented parallelization procedures.

The third tool belonging to the category of *de novo* assembly tools, SPAdes, scales best among the *de novo* assembly tools but also showed a good scaling performance compared to all other benchmarked applications. In the beginning, the runtime is almost halved by doubling the cores, but this behavior stops with the step from 8 to 16 cores. But with each core the runtime can be decreased by two to three percent as observed for the long dataset on a bare-metal machine. There is still a gain in performance, also for higher CPU core numbers, that the other assembly tools in this study did not show. A reason for this specific performance loss could be the change from a single NUMA (Non-Uniform

Memory Access) node to an additional NUMA node, where the inter-process communications are slower than on a single NUMA node. In general, SPAdes shows high wall-time values. This can be explained by the design of the algorithm, which iterates over different sizes of k-mers. As SPAdes uses k-mers with different lengths to optimize the assembly, each step needs nearly the same amount of time. Reducing the number or range of k would lead to lower wall-times. As no effort was put into the evaluation of the quality of the results, the decision for the selected *k's* was left to SPAdes.

TensorFlow, or more precise the CIFAR-10 application, mostly showed a good scaling behavior. The training workload can be easily divided as the intermediate computations are not depending on each other. Further, the TensorFlow framework is tuned to a high performance. Nonetheless, the wall-time values are almost halving in the beginning. With the usage of more than ten cores this behavior stops. The runtime is still decreasing but not as strong as before. This behavior is shown over all environments and all numbers of applied steps. So far, no plausible explanation for that behavior was found As this behavior has been observed independently from the environment and the size of the applied steps, it can be assumed that something on the TensorFlow or CIFAR-10 code level or on the NUMA topology level might be a reason for this.

The most noticeable behavior shows GROMACS. Up to a usage of 16 CPU cores everything is fine and the runtimes are almost halved by doubling the number of cores, leading to a good scaling performance. Unfortunately, there are inconsistencies, interrupting the linear trend, caused by measured runtimes of around one second, meaning that GROMACS did not start the simulation. The log files reveal that the chosen number of CPU cores can not be applied to the present molecule due to the resulting domain decomposition. The reason for that behavior is that the GROMACS developers have put lots of effort into performance tuning. A core piece is the built-in domain decomposition algorithm to apply dynamic load balancing. The following explanation was deliberately kept short and simple in order to not exceed the scope of this work. A more detailed description can be found in the original paper [205] and [206] for the updated version. The molecule, or more precisely the simulation box, is divided into different cells. These cells need to communicate with each other based on the cutoff radii for bonded and non-bonded interactions. This means, if cell 0 is the starting point, it will be checked which other cells are interacting with cell 0 and need to communicate with it. These interactions define how the simulation load is distributed on the processors. The behavior observed in this study can be explained by the fact that all of the failed CPU core configurations are larger prime numbers by itself, like 29 or consists of larger ones like 34 (2 times 17). Depending on the molecular system, such prime numbers chosen for the number of threads can end in very sharp and thin cells resulting in problems to simulate interactions correctly. Therefore, GROMACS performs a check to see whether the chosen thread number is appropriate or not. If not, the simulation will be stopped at the beginning, resulting in the measured one second wall-times. For other molecular systems than the one used in this study, this can change and the not working CPU core configurations could work. However, the newer versions of GROMACS no longer use the concept of charge groups. They have been replaced by pair lists using a Verlet buffer containing the interactions of particles, or more precisely particle clusters that are constructed by spatial gridding in two dimensions and spatial binning in the third dimension. The outcome remains the same using prime numbers as thread numbers can lead to very thin and sharp cells or very large cells with too many interactions, that could not be simulated efficiently.

4.6.3 Generalizability

Throughout this work, several aspects of the scaling behavior of the different applications, datasets and execution environments have been discussed, including the possible overhead of the different environments. First, the presented results are valid for the selected tools, datasets and environments. However, for specific findings a general statement can be made. Looking at the different datasets and the resulting runtimes, no changes regarding the scaling behavior were noticed. Therefore, the size of a dataset seems to have no effect on the multithreading abilities of a tool, assuming a tool behaves the same, no matter the input size, as seen for Clustal Omega where parts of the algorithm are only activated if a certain input size threshold is passed. But in general, it seems to be true that datasets have no influence on the scaling behavior. With this knowledge, it would be possible to first run performance evaluations on smaller datasets and save time on larger ones. The same statement can be made related to the different environments. There seems to be no effect on the scaling behavior, but of course an overhead produced by additional layers will always exist. Another result that can be transferred to other tools and applications are results regarding the used programming languages and the corresponding implementation. As already mentioned in Section 4.6.1, for the CIFAR-10 TensorFlow application and GROMACS, the performance difference between a bare-metal environment and a virtual environment is rather high, compared to other tools, as the implementation of both tools is based on programming languages (C or C++) that are able to map their instructions efficiently to machine instructions. This benefit will get lost if one or more translation layers are in between. Further, it is possible to generalize this result. The closer a tool is implemented to machine instructions, the higher is the performance loss if virtual environments are used, especially with regard to the single core performance.

A generalized statement based on algorithms or data structures is more difficult. Algorithms are used to solve a given problem or to find an approximate solution. The problems to be solved are different for the selected tool categories. For the category of sequence alignment, all tools showed a sufficient scaling behavior. The same is true for the CIFAR-10 application and GROMACS. All of the underlying problems can be parallelized very well, especially for the category of sequence alignment. For the multiple sequence alignment category it looks different, as well as for the assembly tools. In both categories, tools are struggling to show a scaling performance close to linear. This suggests that due to the nature of the problem, parallelization is difficult to achieve. This is also the case if similar data structures are used, like for the selected assembly tools (Velvet, IDBA, SPAdes). For Velvet, only small parts of the implemented algorithm are parallelized. For IDBA, the benefit of multiple cores stops with 16 cores, whereas SPAdes shows a satisfying scaling behavior. Based on the data structure only, no general conclusions can be made, it still depends on the algorithm and the implementation. A look at the tools using similar algorithms like Bowtie2 and BWA or MAFFT and Clustal Omega reveals that some general assumptions can be derived. Both pairs of tools behave similarly regarding their scaling properties, especially Bowtie2 and BWA are very close in terms of their algorithms and data structures. Therefore, it is possible for tools to behave the same but in the end it still depends on the implementation.

4.7 Conclusions and Future Work

Based on the collected results, it can be said that virtual environments cause an overhead, that can not be neglected. Tools designed to make use of bare-metal hardware features might experience a stronger performance loss than tools implemented in a more abstract design. So it needs to be decided for each application whether a generated overhead is acceptable but linked to a gain of flexibility or the other way round. Parallelization is a good way to speed up processes but unfortunately not every of the tested tools offering parallelization through multithreading scales with larger numbers of CPU cores. It depends strongly on the underlying problem whether it can be divided efficiently into smaller subproblems or not. A further strong limitation can result from the implementation of multithreading libraries, as this study showed. Therefore, algorithm and implementation have to fit well together.

One outcome of this study is that the required time to generate the index structure of a reference genome with Bowtie2 can have large differences. Since a reference index structure is usually built only once, most users will probably not even notice. Instead, the pairwise alignment procedure, also performed in this study by Bowtie2, shows a nearly perfect scaling behavior and therefore takes huge advantage of multiple CPU cores. This is also true for the other sequence alignment tools, BBMap and BWA. Other tools like Velvet seem to gain no advantage from multiple available cores or are limited to a supported number like IDBA-UD. As other tools like SPAdes using a similar data structure scale well, there seems to be a limit due to the implementation. As mentioned first, the used tools are mostly not affected by the chosen datasets, except for Clustal Omega. For Clustal Omega the number of sequences can make a difference, but larger datasets should benefit from the usage of multiple cores, as the additional benchmark run showed. Tools that have been proven to be robust and very scalable are TensorFlow, the CIFAR-10 application in particular and GROMACS. The scalability of both tools benefits mostly from a well defined decoupling of the main problem into smaller subproblems, which is a basic requirement to achieve a high scalability.

It is planned for the future to extend BOOTABLE by more applications and cover even more research areas. Especially the topic of GPU accelerated tools in the area of bioinformatics seems to be promising to get deeper insights, also regarding virtual environments and scaling effects. Further, more datasets should be added and more suggestions on the types of datasets that can have an impact on different tools, as seen for Clustal Omega, should be provided. Finally, it would be beneficial to build a public database of already benchmarked tools and curated recommendations about their reasonable resource usage and constantly adding new results to make the life of users and resource providers a bit easier. It is not always the more the merrier.

Chapter 5

Virtual Cluster

5.1 Motivation

Usually, no batch system or scheduler is available by default In a cloud environment like in an HPC environment, because the working principle of compute clouds is mainly based on single virtual machines [85]. VMs may include multiple virtual CPU cores but these cores are typically not distributed over different machines, as it would be the case for a compute cluster. To make proper use of available cloud resources, application developers possibly need to cloudify their application in some way [207]. The cloudification process requires time and effort, depending on the application. But the effort might be worth it as a cloud environment mostly offers more flexibility than a classical compute cluster environment. Further, it is precisely this flexibility that enables a cloud environment to emulate a compute cluster, even if it does not reach the same performance values [208]. If an application is not that critical, regarding network latency, which might be the case for applications using a message passing interface (MPI) to distribute the workload over multiple nodes, it is possible to run the same workload on a virtual cluster. A virtual cluster has the same structure as a bare metal compute cluster but components like compute nodes and network are mostly virtualized.

Another upcoming demand that can be observed is the handling of sensitive -omics data [209]. Sequencing technologies are getting cheaper and are integrated in the daily work of hospitals, for example [210]. The collected data are used in different ways, for example in the field of personalized medicine using specific genetic information of patients for a tailor made therapy or for larger studies of rare diseases. But all data related to real persons, especially genomic data, need to be highly protected, required by law. To analyze larger amounts of sequencing data computing resources larger than a common workstation might be required. To analyze such data, larger virtual machines or a virtual cluster in a cloud environment might be as suitable as a classical compute cluster, depending on the application. One problem with sensitive data is, that they should be only accessible by people they have been granted access to it [209]. In a usual, classical compute cluster environment a file system is available, that is shared between all compute nodes to enable data access among compute nodes. But this so-called scratch or work file system is not specially protected. If other users are also using the same HPC cluster they might be able to access sensitive data even if they are not allowed to, as strict separation is difficult in this setup [211]. Although there are concepts in place handling permissions via user ids or working with access control lists the separation might still not be sufficient. This problem could be solved if a dedicated cluster is set up for the purpose of analyzing sensitive data and assigned to a group of authorized users. If such a cluster is shared with other groups, it needs to be ensured that no data or fragments of it are accessible by unauthorized users. This would also involve the development of a clean-up strategy. Furthermore, this option involves the purchase, deployment and maintenance of a compute cluster, which can be costly and time consuming. The usage of an already existing cloud environment could lead to a better separation, as most clouds are designed to handle multiple users. Usually they are separated in a management structure based on regions, zones, projects or down to a single user [212]. Especially the storage component can be separated from other users as there does not directly exist a shared storage space everyone has access to by default. If a compute cluster environment is required to conduct the desired data analysis, due to application reasons, this could be emulated by using multiple VMs internally connected to form a virtual cluster. Such a virtual compute cluster and also the shared storage component can be assigned to a single project and separated from other users. A virtual cluster approach combines the advantages of a cloud environment regarding the flexibility, with the power of an HPC cluster to use more resources than a single node or VM could offer [208]. Even if it is virtual, a computing cluster needs some care starting from the deployment to its runtime and deconstruction.

To make these processes easier, this chapter presents an automated virtual cluster deployment tool for OpenStack based clouds named VALET (Virtual UNICORE Cluster). The developed deployment tool is easy to use and comes along with the setup of a computing cluster including a batch system (PBS TORQUE), shared file system (BeeGFS/BeeOND), middleware with a workflow engine (UNICORE) and a monitoring system (Zabbix). The deployment is handled by Terraform that is especially designed for the management of infrastructures as code. It is possible to add new nodes to an existing cluster or to remove them, without any downtime. In addition, a meta scheduler has been implemented, that can be used to handle the up- and downsizing of a cluster automatically, depending on the measured load. The following sections illustrate the used software components and their connections. Further, workload simulations are presented to evaluate the implemented scheduling algorithm regarding its capabilities.

5.2 Methods

5.2.1 Components

File System

For the shared file system, the BeeGFS based BeeOND (BeeGFS On Demand) [213] tool has been chosen. BeeOND is a wrapping tool to create one or multiple BeeGFS instances on the fly. The prior use-case of BeeOND is to connect a number of compute nodes in an HPC cluster, take a local hard-drive space of a node and span a temporary work file system over the participating nodes for the duration of a compute job. Another use-case, proposed in this paper, is the provisioning of a shared file system. Other representatives of shared file systems are for example NFS (Network File System) [39] or in the context of an OpenStack cloud environment, Manila [214]. Due to the modularity and the resulting flexibility of BeeGFS in interaction with BeeOND, it is possible to create a shared file system across different virtual machines. Further, BeeGFS offers the possibility to add and remove storage servers, in this case VMs, participating in the virtual cluster without any downtime of the shared file system. This ability is important to utilize the flexibility of a cloud environment and in particular for a virtual cluster. Through the dynamic starting and stopping of VMs as computing nodes, it is possible to respond to the currently required resources. To use BeeOND and the underlying BeeGFS in a virtual cluster environment, some functionalities which are explained in Section 5.2.2 had to be implemented.

Batch System

As resource management system (RMS), PBS TORQUE is used. The Terascale Open-source Resource and QUEue Manager (TORQUE) is a community driven project based on the portable batch system (PBS). Criteria for the choice of TORQUE were the lightweight installation process and the possibility to add and remove computing nodes to or from an existing cluster without any downtime of the batch system, which can be fulfilled with some effort. Further systems like Slurm or various flavors of the Sun Grid Engine have been evaluated, but the lack of specific features or the discontinued development lead to the decision to use TORQUE. Furthermore, TORQUE is compatible with lots of other tools in the context of cloud and HPC, for example the middleware UNICORE [103], Galaxy [215] or pipeline systems like Nextflow [216]. It therefore provides good extension possibilities for future use cases.

Middleware

On top of the compute cluster environment consisting of BeeOND as file system and TORQUE as batch system, the middleware UNICORE is integrated. UNICORE is a middleware software which comes with a workflow engine. It is developed at the research center in Jülich and by further partners. The UNICORE software stack offers a number of different components to handle HPC environments in a simple way for both, administrators and users. Furthermore, UNICORE provides a web-interface (UNICORE-Portal) for user interaction. The resources handled by the UNICORE server and workflow component, can be accessed by the UNICORE Rich Client (URC), an advanced graphical user interface, or the UNICORE Command Line Client (UCC). UNICORE is made to hide a complex compute cluster structure and it presents a simple but powerful job submission system to the outside and eases the user experience. UNICORE is written in the programming languages Java and Python and therefore platform independent.

5.2.2 Implementation

Cloud Platform

OpenStack has been chosen as cloud-platform management system. OpenStack is a widely used open source cloud-framework, initially started by the company Rackspace and the NASA. As my institute is a partner of the de.NBI Cloud providing cloud resources running on OpenStack, it was the most obvious choice. Other options like AWS, Azure or Googlecloud would have been possible, but were postponed until experiences have been gained with the OpenStack targeted development. Further, it was possible to mobilize a broader user community as test candidates to get feature requests more often and to find bugs faster.

BeeOND

To use BeeOND and the underlying BeeGFS in a virtual cluster environment, two functions have been implemented: (1) The unmodified BeeOND version assumes, that it will be used in a HPC environment and therefore all nodes can access each other in a password-less, non-interactive way. But the access in a cloud environment to a VM is handled by SSH-Keys, which is a behavior that was not covered yet by BeeOND. So, an additional option to the command line parameters has been added to enter a path to a SSH-Key and to handle the add and remove procedure of VMs. (2) It was required to add another option specifying the login user of a VM. The default behavior of BeeOND is to take the current user name of the shell starting the BeeOND tool. But this does not have to necessarily be the user of the connecting VM. The login user of bare cloud images, which are taken as a kind of template for new VMs, are named after their operating systems. For a CentOS system, the login user would be centos and for an Ubuntu driven system it would be ubuntu. If a master node runs on CentOS but compute nodes run on Ubuntu, the start of the BeeGFS file system did not work as it was not intended for this kind of application. Therefore, the possibility to change the login user name in connection with the handling of the SSH commands has been implemented. Due to these two extensions of the original BeeOND version, it is now possible to create and deconstruct a BeeGFS file system on the fly over several VMs in an OpenStack cloud environment integrating attached OpenStack Cinder Volumes as storage medium. To enable dynamic scaling of a compute cluster, functions for adding and removing individual VMs to and from an existing BeeGFS file system have been implemented. Parts of the original BeeOND implementation have been extracted and modified to handle the add and remove procedures for single VMs. The modifications involve the usage of SSH Keys and different login names and the adaption of the start and stop procedures to single nodes for an already set up file system. A detailed description of the individual steps is given in Section 5.2.2.

Deployment

The deployment of the virtual cluster is managed by the Terraform software tool. All required and adaptable variables are stored in a variable file and can be adapted to the specific needs of a user. The deployment is started using the OpenStack API user credentials. Already pre-configured VM images, based on CentOS 7, are available via a local S3 Storage and are uploaded to the associate user project during the virtual cluster deployment process. The initial deployed cluster consists of one master node and two compute nodes. Further, the initial cluster setup can be deployed with different network settings. It is possible to use public IP addresses for all nodes (master and compute), but it is also possible to use only one public IP for the master node as entry point and use internal IP addresses for the compute nodes. The last setup requires a second interface for the master node to communicate with the compute nodes, but saves valuable public IP addresses. An illustration of the second setup is shown in Figure 5.1.

In a bare metal cluster the nodes are usually connected directly over a network and especially the compute nodes are usually not accessible from the outside so they can communicate without password requests by exchanging key pairs. For a virtual cluster, a similar setup can be established. The difference is that the private key needs to be placed on the master node according to the public key of the compute nodes to make communication possible. Per default, the master VM is accessible from anywhere as it owns a public IP address. Therefore, it is no option to use the same key for the compute VMs as for the master VM. That is why a second key pair for the compute nodes is created automatically during the deployment process. This key pair will only exist as long as the cluster exists.

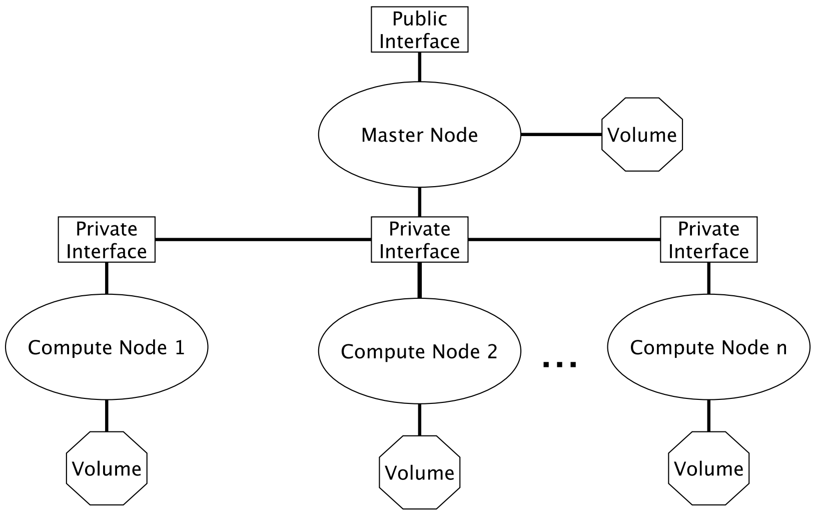


Figure 5.1: Schematic representation of the network setup using an internal network (private interface) for the compute nodes.

Thus, the private key of a user is never exposed on one of the nodes externally. The workflow of the deployment process is illustrated in Figure 5.2. After all VMs are deployed and available, developed scripts written in bash are executed to install and configure the virtual cluster, starting with the file system (BeeGFS). Afterwards TORQUE is configured and started. If the setup of the shared file system and the RMS is finished, the basic setup is available. On top of this basic setup, the middleware (UNICORE) is configured. As a final step, the monitoring (Zabbix) is started. After the initial cluster is deployed, the whole structure can be managed by Terraform. The deployed virtual cluster resources are available as infrastructure as code. This code structure allows to keep track of the virtual cluster status and enables possibilities to add and remove components, which are described in Section 5.2.2.

General Up- and Downsizing

The automated deployment of a virtual cluster is nice to have, but to make use of the flexibility offered by a cloud environment a general up- and downsizing mechanism has been implemented. Again, Terraform is used to add and remove nodes to and from an existing cluster. This can be done manually with the implemented add and remove procedures.

If more resources are needed, an additional node can be started. Terraform uses the OpenStack API to start a new VM from the already created compute node image by the initial cluster setup. Further, it is possible to deploy new nodes with different kinds of flavors and therefore other resources than the already existing nodes. This adds an extra level of flexibility to the implementation.

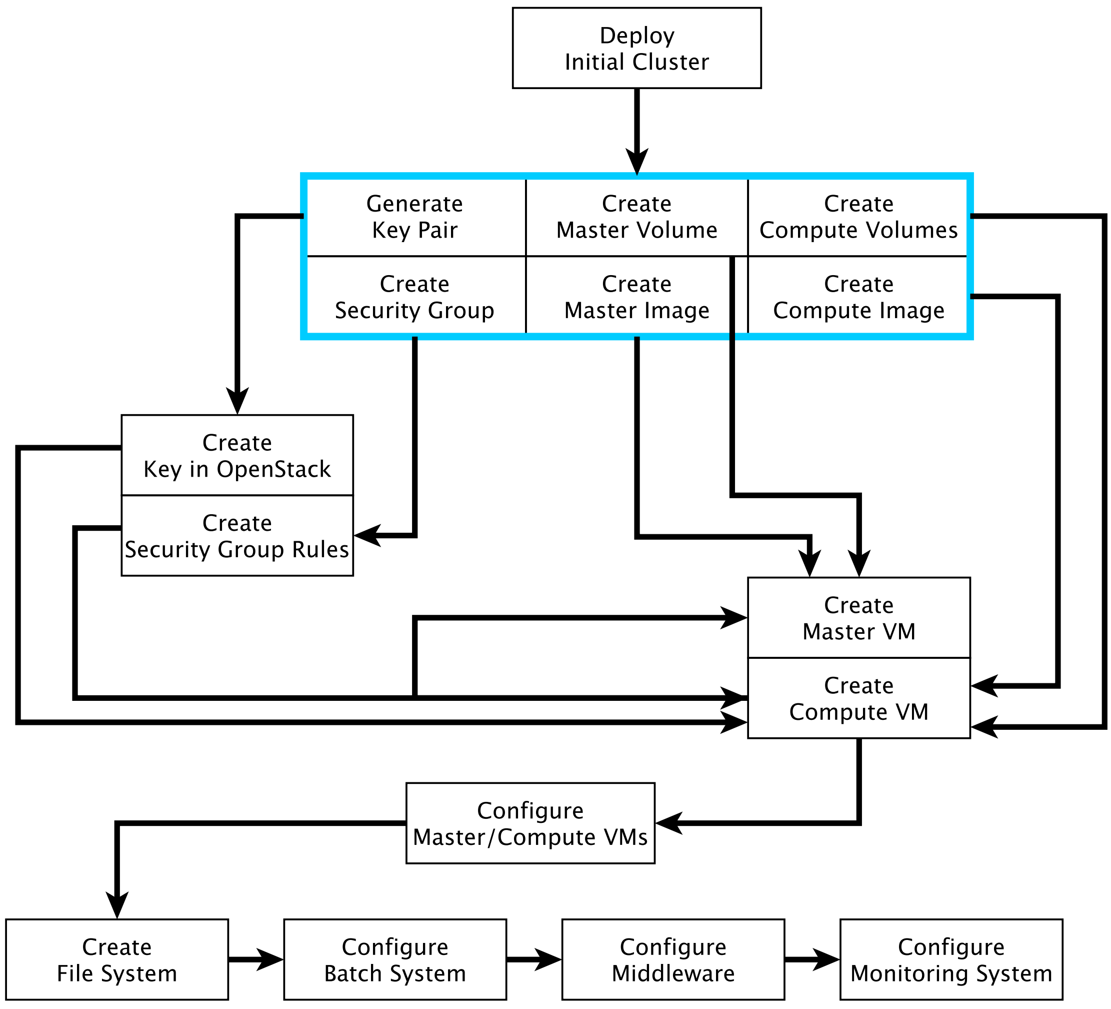


Figure 5.2: Flow-diagram illustrating the deployment process of the initial virtual cluster performed by Terraform. The blue framed processes have to be executed first.

After a new VM is fully available, it needs to be integrated into the existing cluster. The integration involves four main components. First, a new node needs to be integrated to the already existing shared file system, where other nodes of the cluster are part of. Second, the batch system needs to become aware of a new node as additional resource. Third, the resources UNICORE can use, need to be updated to keep the resources consistent between the batch system and UNICORE. Finally, a node is added to the monitoring system. The integration of a node into an existing file system is achieved by extending the available BeeOND implementation with the feature using specific user names and paths for the SSH connection from the server node to the client nodes. Further, all unnecessary procedures of the initial BeeOND start tool have been removed as only a new storage, metadata and client process needs to be started on the added node. To achieve this, a temporary description file is created. The description file contains the internal IP address of a master node and a node to be added. The modified BeeOND start-up script starts all required components on a new node and adds it to the already existing file system. As the file system needs to be available on a new node before the batch system can use it, this has to be done first. In the second step, a new node is included by the batch system (PBS TORQUE). This is achieved by configuring the client services on an added node and by starting the services. After a new node has been registered by the master node, it can be used directly. The resource numbers are detected automatically. After the new resources are made available to the batch system, the resources available for UNICORE need to be updated as well. This is done by changing the corresponding configuration file entries. The last step is to include the new node into the monitoring system, which is done by installing, configuring and starting the according Zabbix agent on the new node.

Due to the fact that Terraform encapsulates the state files of managed infrastructures in single directories, a second, intermediate Terraform environment is initialized and used to deploy additional VMs. After a new node (VM) is completely deployed, it needs to be integrated into an already existing Terraform code structure, holding the status information of the remaining cluster. This is necessary to manage a whole virtual cluster at once, not only single parts. To achieve this, the state information of a new VM is moved to the existing cluster state file using Terraform's state move function. The old state information is cleaned up to reuse the intermediary Terraform environment for subsequent start procedures.

Beneath the possibility to add nodes, it is also possible to remove nodes from an existing cluster. The remove procedure handles the same four main components as for the add procedure, but the order is slightly different. First, the batch system blocks the scheduling of new jobs on the chosen node and waits until all jobs, which are still running on that node, are finished. After that, the node is removed from the batch system. Afterwards the node needs to be removed from the file system. As the data on the shared file system are distributed over all participating nodes due to the concept of the underlying BeeGFS file system, this needs to be done with care. First, the storage node is blocked, so that no new data will be written to it. After the node is blocked, all residing data will be transferred to the left nodes, spanning the shared file system. However, this is only done if the now decreased total storage size is sufficient to hold the data to be transferred. If the left over space is not sufficient, the process will be stopped, otherwise the data are migrated. After the migration step, the node is removed as storage node and all processes belonging to the file system on it are stopped. The original BeeOND stop-procedure has been modified to only stop the chosen storage node and not all participating nodes. In the end, the node to be removed is deleted from the Zabbix monitoring system and the UNICORE resource entries are adapted. Finally, the VM is destroyed by Terraform using the OpenStack API. Again, it needs to be assured that the state of the cluster is consistent with the actual running cluster. Therefore, the Terraform destroy function is used in combination with the target function to remove only a specific node from the complete structure and not to destroy the whole cluster. The coded state structure of the cluster is now consistent with the structure of the present cluster.

All steps necessary to resize a virtual cluster are done without affecting the already running jobs and processes, achieving it with zero downtime. The resizing of a cluster can be done manually, executing the implemented start and stop procedures. But often jobs or whole pipelines can run multiple days or even weeks, also with different numbers of jobs and resource usage, which can not be handled manually all the time. To automate the resizing of a cluster, a resize algorithm has been implemented, measuring the current cluster utilization within given time intervals. Based on the measured utilization, connected to different thresholds, a resize process can be triggered. The resize algorithm is explained in detail in Section 5.2.4.

5.2.3 Automated Scaling

This section describes the implemented algorithm and the resize mechanism. At first, the general structure is explained. The resize mechanism is separated into two parts. One part is working on the master node of a cluster, the other on the desktop machine of a user. Both parts are explained in the following subsections. The general structure is illustrated in Figure 5.3.

Cluster

On the cluster side the built-in systemd service, adopted by the major Linux distributions, is used for the periodic execution of the resize algorithm. The resize procedure is triggered in a defined time interval, set to five minutes per default, by a systemd timer, illustrated with (1) in Figure 5.3. This implemented

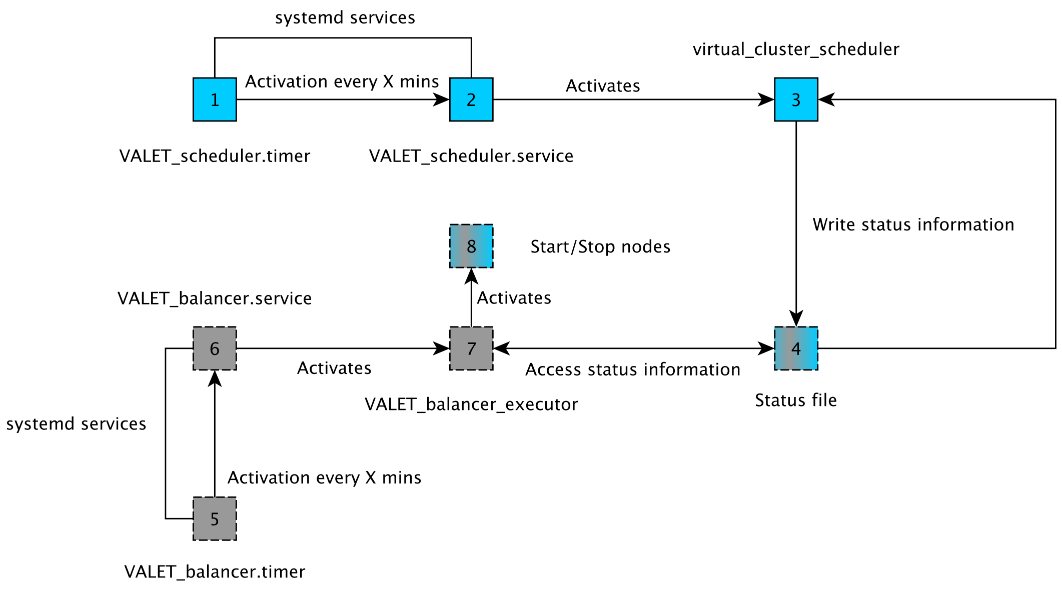


Figure 5.3: Illustration of the cluster resize structure. The steps from 1 to 3 are running on the master node of a cluster (blue boxes with solid lines). Step 4 belongs to the cluster side, but the desktop side interacts with it. Steps 5 to 7 belong solely to the desktop side (gray boxes with dashed lines). Step 8 is residing on the desktop side but is interacting with the cluster side as well.

timer triggers the corresponding systemd service (2), starting the resize algorithm (3). After the resize evaluation is finished, a state parameter is set to inform the services, running on the desktop side, to decide which action is performed (4). Possible actions are: staying with the current cluster configuration, start a new node and add it to the cluster or remove a node from the cluster. The decision for these actions are made by the resize algorithm based on the gathered cluster usage information. The services running on the desktop side process the committed parameter and initialize the necessary actions.

Desktop

The desktop services are running on a workstation of the corresponding user applying a virtual cluster. The workstation can also be a cloud VM but then the already explained problem of an exposed private key would come into play. The structure is similar to the cluster side. A pair of a systemd timer and a systemd service represented by (5) and (6) in Figure 5.3 is used to trigger the execution procedure (7) that consumes the state parameter of the resize algorithm (4). The default cycle time is set to one minute. This means that the resize execution procedure checks the status parameter set by the resize algorithm on cluster side every minute and executes the appropriate actions like adding or removing nodes if necessary.

5.2.4 Scaling Algorithm

The implemented algorithm to resize a cluster according to its utilization can be split into two different behaviors. One behavior is to start a new node directly if specific criteria are fulfilled. The other behavior is to collect different weight values over time until given thresholds are reached that can trigger a start or remove procedure. These two options make it possible to react directly to a high utilization and to add more resources to a cluster or to increase the resources over time if no direct events are triggered but the available resources are fully in use and the running jobs might benefit from more resources. To measure the utilization of a cluster and to decide, whether an increase or decrease of resources is necessary, different weights and thresholds are used. The default values are shown in Table 5.1.

To ease the representation of the resize algorithm, it is divided into five parts and will be illustrated by corresponding pseudo codes (Algorithm 1 to Algorithm 5). In general it needs to be differentiated between actions (start, stop) that are triggered immediately based on a specific event or actions triggered by accumulated events over time. The first part of Algorithm 1 decides if a node needs to be started, stopped or nothing has to be done, based on the collected weight values over time. If the sum of different weight values is larger or equal to the set threshold (start_threshold), the start parameter string is written to

Name	Value	Category
W1	0.1	W
W2	0.3	W
W3	0.3	W
W4	0.2	W
W5	-0.2	W
qrr	0.4	T
rfr	0.8	T
rfd1	dyn.	T
rfd2	dyn.	T
rfd1_multiplier	10	M
rfd2_multiplier	5	M
rcCr	0.2	T
cnc	2	T
mnc	4	T
start_threshold	1.0	T
stop_threshold	-1.0	T

Table 5.1: List of variables and thresholds, used by the resize algorithm, with their corresponding values and categories weight (W), threshold (T) and multiplicator (M). The variables rfd1 and rfd2 are calculated dynamically.

the status file. The same holds for the stop procedure with the difference that negative numbers are used. Of course it will always be checked that the start or stop of a node is generally possible, according to the values of the minimum number of nodes (cnc) and the maximum number of nodes (mnc). If none of the thresholds is passed, the algorithm continues.

The main part of the algorithm starts with the distinction between the case of zero running jobs or zero queued jobs. This information points to a cluster that is not utilized at the moment as no jobs are running or jobs are still running but currently no ones are waiting. It needs to be checked for both scenarios, whether the cluster could be downsized or not. If no jobs are running, no jobs should be in the queue, the cluster is empty and the value of weight W5 is added to the start or stop file. If enough negative weights are collected, as shown in Algorithm 1, a cluster could be scaled down. For the second case where no jobs are queued but still running, it needs to be checked how large the utilized resources (number of cores) are compared to the overall available resources. To monitor this, the ratio between the used cores and the available cores (running CPU capacity ratio) is calculated. If the used cores are less than 20% of the overall resources the weight W5 is added to the start or stop file as the cluster seems to be close to idle. Further, the stay parameter string is written to the status file as no immediate action needs to be triggered. The pseudo code of this part is shown in Algorithm 2.

If neither the queued jobs nor the running jobs are zero, the cluster is utilized and it needs to be checked how much of the available resources are used and if adding new nodes would help to reduce the number of waiting jobs. To estimate whether the cluster is overloaded and needs more resources, the ratio between the

```
Data: Start and stop files holding weight values
Result: Decision whether new node is started, stopped or nothing
if start file exists then
    sum up collected weight values;
    if sum of weights $\geq$ start threshold then
        set start parameter;
        delete start file;
    end
    else
        print: threshold not reached;
    end
end
else if stop file exists then
    sum up collected weight values;
    if sum of weights $\leq$ stop threshold then
        set stop parameter;
        delete stop file;
    end
end
```

Algorithm 1: Check sum of collected weights

```
Result: Decision if W5 is added as cluster looks idle
if queued jobs = 0 OR running jobs = 0 then
    set stay parameter;
    calculate ratio of used cores to available cores;
    if running jobs = 0 OR ratio < rCcr then
        add W5 to start or stop file;
        exit;
    end
    else
        print: cluster is still in use and not underutilized, nothing else to
          do;
    end
end
else
    check further algorithm steps (Algorithm 3);
end
```

Algorithm 2: Check if downgrade is possible

number of queued jobs and running jobs is calculated first. The calculated value is compared to the defined threshold. If the value is equal or larger than the set default threshold (qrr), 40% or more jobs regarding the number of running jobs are in the queue. That is a first hint that adding a node might help to reduce the waiting time. To get a more detailed insight into the wall time of compute jobs, the mean wall times per job of the finished jobs and the currently running jobs are calculated. At the end, a ratio of these mean time values is calculated. Further, a prediction value is calculated representing the time it takes for all queued jobs to finish based on the mean wall time of already finished jobs by multiplying the number of queued jobs with the mean wall time of finished jobs. Subsequently, the threshold values for the variables rfd1 and rfd2 are calculated by multiplying the mean wall time of the finished jobs with their default multiplication values (rfd1_multiplier, rfd2_multiplier). The corresponding pseudo code is shown in Algorithm 3.

```
Result: Ratios to measure the cluster activity
if queued running ratio $\geq$ qrr then
    calculate mean job time of finished jobs;
    calculate total finish time of all queued jobs;
    calculate rfd1 threshold;
    calculate rfd2 threshold;
    calculate mean job time of running jobs;
    calculate ratio of mean running to mean finished job wall times;
end
check further algorithm steps (Algorithm 4);
else
    check further algorithm steps (Algorithm 5);
end
```

Algorithm 3: Calculate different values to measure the cluster activity based on finished and running jobs.

The calculated ratio of running and finished wall times is compared to a threshold of 1. If it is larger than 1, this means that the mean wall time for the currently running jobs is larger than the mean wall time value of already finished jobs. If jobs have similar runtimes, that could mean that the running jobs might be nearly finished and new jobs can be scheduled in the near future. The first assumption is that the jobs have similar wall times and there will be some free slots soon. But it can also be the case that the wall time of jobs changes and takes longer than already finished ones. Therefore, the value of weight variable W2 is added to the start file. The default value of W2 (0.3) is quite high compared to the other variables and to the default threshold value of 1.0. This should ensure that a cluster can scale up quickly even if heterogeneous jobs are submitted, for example pipelines with rather short jobs in the beginning and longer lasting jobs afterwards.

To prevent such situations, it needs to be checked whether the ratio of running and finished wall times is in the range between 1 and 0.8. This means that the mean runtime of currently running jobs is close to the mean runtime of already finished jobs. It can be expected that there will be some free resources soon. In order to make a decision, it is additionally checked whether the total finish time of all queued jobs is less than or equal to the dynamically calculated threshold rfd1. If this is the case, it is an indicator for a small number of waiting jobs and it might be unnecessary to start and add a new node to the cluster. To address this case, the weight W3 with the same value of W2 is added to the start file. If the total finish time is larger than rfd1, a new node is started directly as this indicates the exceeded acceptance level of the waiting time. This acceptance level can be adjusted by a user choosing high values for the rfd1_multiplier parameter, to be more conservative regarding the start of new nodes or choosing low values to add nodes more aggressively.

In case of the ratio of running and finished wall times is even lower than 0.8 and jobs might not finish soon, it is checked whether the predicted overall runtime is higher than the threshold given by rfd2. The multiplier of rfd2 is chosen lower per default than for rfd1, as in this part of the algorithm it is already known that additional resources can reduce the time until all jobs currently queued are finished. If the predicted time is higher than the value of rfd2, a new node is started directly. Even if the predicted time is lower than rfd2, the weight W4 is added to reward the fact, that jobs are waiting. The pseudo code illustrating this part of the resize algorithm is shown in Algorithm 4.

The algorithm closes with the case that the queued running ratio is smaller than the chosen default ratio of 0.4, illustrated by Algorithm 5. In this step, the stay parameter is written to the status file as no direct action will be executed. In this part, it is known that the available resources of a deployed cluster are not used to its limits but the demand of resources can increase. Therefore, weight W1 with a value of 0.1 is added to the start file to decrease the values required to reach the start threshold value. This procedure allows it to scale up the cluster in a shorter amount of time with additional weight values.

5.2.5 Workload Simulations

To check the functionality of the implemented resize algorithm simulations were performed. The simulations were conducted on a cluster deployed with VALET, using resources from the de.NBI Cloud site Tübingen with a maximal number of nine compute nodes with 8 CPU cores, 16GB RAM each, and one master node. The time the scheduler history is stored has been set to 36000s (10 hours) as this turned out to be a suitable starting point after the first tests. Further, the weight W5 has been set to -0.2 and the according threshold to -1. To test

```
Result: Decision whether nodes are started or weights are added
if queued running ratio $\geq$ qrr then
    check prior algorithm steps (Algorithm 3);
    if running finished ratio $>$ 1.0 then
        add W2 to start file;
        exit;
    end
    else if running finished ratio $\geq$ 0.8 then
        if predicted finish time $\leq$ rfd1 then
            add W3 to start file;
            exit;
        end
        else
            set start parameter;
            exit;
        end
    end
    else if predicted finish time $\geq$ rfd2 then
        set start parameter;
        exit;
        else
            add W4 to start file;
        end
    end
end
else
    check further algorithm steps (Algorithm 5);
end
```

Algorithm 4: Check running and finished mean time ratios to decide on adding weight values or to start an additional node directly.

```
Result: Add weigth W1 to start file
if queued running ratio < qrr then
    set stay parameter
    if start file exists then
        add W1 to start file;
    end
    else
        create new start file;
        add W1 to start file;
        delete stop file;
    end
end
```

Algorithm 5: Check whether queued running ratio is below the threshold (qrr), to add the value of weight W1.

the behavior of the explained resize algorithm for different kinds of workloads, a real world example has been implemented. The pipeline for this as well as the expected runtimes of the individual steps were provided with the kind support of Peter Ebert [217]. The pipeline has been emulated using the sleep command for the sake of simplicity. To simulate the dependencies between the different steps, the job dependency feature provided by PBS TORQUE has been used. This ensures that the different steps of the simulated pipeline are executed in the same order as for the original pipeline. Usually, real world examples do not have the same runtime even for the same data. It was thought about simulating such a kind of behavior using randomly chosen wall times in the possible range for each submitted job. This brings some kind of unpredictability. Further, it is possible to vary the number of submitted jobs for each step of the pipeline to get more information about the behavior of the resize algorithm for more divers job scenarios. However, to get comparable results, static values for the wall time and the number of the different jobs were set. A schematic illustration of the pipeline is shown in Figure 5.4.

The pipeline simulating a full genome sequence analysis pipeline has been implemented in a modified way. For the wall times, the original numbers were taken, using the sleep command, but the required compute cores have been scaled down according to the available node specifications. The steps and related resource information are illustrated by Table 5.2. The scaling algorithm has been tested keeping all parameters fixed, except for the interval parameter, to check the load of the cluster periodically. The test interval time numbers reach from 5 minutes down to 2 minutes with steps of 1 minute in between. Furthermore, every scenario has been repeated five times to reduce the impact of outliers. All runs were conducted using an empty scheduler history and the initial cluster setup, consisting of two compute nodes and the master node.

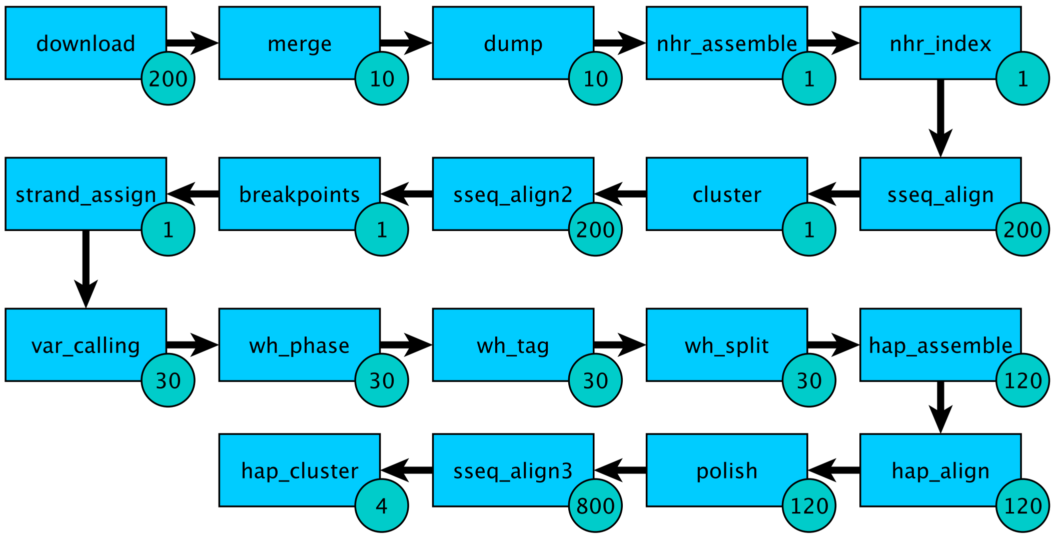


Figure 5.4: Illustration of the real world pipeline used for evaluation purposes. The added numbers indicate the number of individual compute jobs of each step, that can be executed independently. This helps to understand when a larger number of compute nodes can speed up the execution.

Step	#jobs	#cores	wall time [s]
download	50 - **200**	1	300 -3600 (1800)
merge	1 - **10**	1	10800
dump	1 - **10**	1	10800
nhr_assemble	1	8	36000
nhr_index	1	1	4800
sseq_align	50 - **200**	3	300
cluster	1	1	10800
sseq_align2	50 - **200**	3	300
breakpoints	1	8	1800
strand_assign	1	8	1800
var_calling	20 - **30**	8	900
wh_phase	20 - **30**	1	300
wh_tag	20 - **30**	1	300
wh_split	20 - **30**	1	2700
hap_assemble	80 - **120**	8	900
hap_align	80 - **120**	8	480
polish	80 - **120**	4	600
sseq_align3	200 - **800**	3	300
hap_cluster	4	1	900

Table 5.2: List of steps for the real world pipeline example using the original wall times, including the names of the steps, the number of jobs or ranges, used number of cores and the assigned wall time per job. The pipeline has been emulated using fixed job numbers (maximal values in bold) for better comparison reasons. For the download step a variable wall time has also been noted, which has been set to half of the maximal value.

5.3 Results

In this section, the achieved results of the general cluster deployment and of the implemented test scenario are presented. The initial cluster setup consists of one master node and two compute nodes. In total, the initial deployment takes around 9 minutes, the required time for the different steps is shown in Table 5.3. Afterwards, the cluster is completely set up and configured, including the Zabbix monitoring system and the middleware UNICORE with its workflow engine. The deployment process is illustrated in detail in Figure 5.2. Most steps are processed in parallel. In the first part, the security groups and rules, the key pair for the compute nodes and the three necessary volumes are created. In the second part, the required compute and master images are uploaded to the cloud infrastructure. Afterwards, the compute VMs are deployed and the master node finally follows. After all VMs are available, the configuration process is started.

Step level	Description	Time [s]
1	Security group, rules	1
1	Key pair	2
1	Volumes	10
2	Image compute	120
2	Image master	150
3	Compute VM 1	60
3	Compute VM 2	60
4	Master VM	80
5	Configuration	220

Table 5.3: List of the different steps during the initial cluster deployment incluing the requrired time in seconds. Processes assigned to the same step level are executed in parallel.

Further interesting is the time to add and remove nodes from an existing cluster. The time of adding a new node adds up to around 90 seconds but this depends on the size of the created volume and how fast this can be created and also formatted. The 90 seconds hold for a 100 GB sized volume in the stated infrastructure. The deployment of the VM itself is mostly fixed with around 13 seconds as the initially uploaded compute node image is reused. Removing a node is also dependent on the storage component and on the scheduled jobs. If jobs are currently running on the node to be removed, the procedure waits until the node is free. After that, the storage volume has to be removed including the redistribution of the currently saved data to the other nodes. The best case scenario is an empty node without running jobs and no data saved. For that case, the time to remove is around four minutes. The longest step here is to wait until the node has been blocked to stop the writing of new data. The destruction of the resources by Terraform is done in about 10 seconds.

To evaluate the effectiveness of the implemented resize algorithm the collected data regarding runtime, number of used nodes and cores are compared to fully

History	Cluster	Check time [min]	Wall time [min]	SD [min]
No	Full, no scaling	-	2245	-
No	Full, no scaling, 8 nodes	-	2345	-
No	Full, no scaling, 7 nodes	-	2500	-
No	Initial	5	2334	9.62
No	Initial	4	2334	8,29
No	Initial	3	2325	9.49
No	Initial	2	2336	2.97
No	Initial	tuned (3)	2337	-

Table 5.4: Mean wall time values and their standard deviation of the pipline example, grouped by the time interval the status of the cluster is checked and whether the initial cluster was already available (Full) or not (Initial, 2 nodes available). Variants have been tested using smaller numbers of maximal usable nodes (7, 8).

Check	Runtime	Nodes	Used cores	Cost savings
0 (8 nodes)	4.45	-	-	7.15
0 (7 nodes)	11.36	-	-	13.39
5	3.96	23.29	3.48	22.68
4	3.95	23.89	3.40	20.89
3	3.56	24.73	3.09	22.05
2	4.04	26.09	3.33	23.11
3 (tuned)	4.10	27.22	3.80	24.24

Table 5.5: Mean overheads in percentages referenced to the values of a static cluster. The overheads of the different parameters are shown as percentages as well as a cost saving value based on the AWS price calculator with On-Demand Instances and comparable resources. The Cost savings are also stated in percentages, meaning how much cheaper a dynamic cluster would be. Also included are calculations for static clusters, using 8 and 7 as the maximal compute node numbers.

upscaled static cluster consisting of nine compute nodes. First of all the, mean wall time value of an already fully scaled static cluster with nine compute nodes shows the lowest mean wall time values overall compared to the other measured scenarios. Therefore, it represents the best case regarding the maximal resource availability.

The mean wall time values of the different time steps reaching from five to two minutes show no clear decline or increase according to the time interval the load is checked. The smallest mean wall time values have been reached with the interval of three minutes. The intervals two, four and five show slightly larger values. But with increasing values for the check interval, the calculated standard deviation values also increase, pointing towards a higher fluctuation regarding the measured wall times (see Table 5.4). To better understand the behavior of the different scheduler intervals, the number of available compute nodes during the execution of the implemented pipeline is shown in Figure 5.5. An overview of the available and the actually used CPU cores is shown in Figure 5.6.

Furthermore, overhead percentages of different values of interest have been calculated, which are shown in Table 5.5. The overheads according to the wall time are very close to each other within a range of 3.56% to 4.04% compared to a static cluster set up. The mean number of available nodes throughout the whole runtime has been decreased by more than 23%, meaning that a dynamically scaling cluster is using 23% less compute nodes than a static, fully scaled cluster. As less nodes are used during the runtime of the pipeline, also less CPU cores are used by the dynamically scaling cluster. The overhead of a static cluster concerning the CPU core usage can be specified with around 3% to 3.5% over all used check time intervals.

Furthermore, the costs of a static cluster and a scalable cluster have been compared to each other. The comparison is based on AWS prices taken from the calculator offered by Amazon[9]. As reference, an On-Demand Instance of type a1.2xlarge (Linux) has been taken, as it comes with the same amount of resources as the used compute nodes in this work. As an hourly base price AWS states 0.2328$ per instance. The costs have been calculated by taking into account the total wall time and the mean number of used nodes per run. As all of the used interval configurations have reduced the number of nodes needed, the costs have been decreased by a value reaching up to 23% with an interval value of 2 minutes.

Besides the non-scaled variant where all nodes were already available, additional configurations were evaluated where the maximum possible compute node number was reduced. The number was reduced by 1 or 2 compared to the maximum number of 9. As a result, the wall time increased. However, the costs also decreased, that is why these scenarios are relevant to check whether a static cluster with fewer compute nodes might be cheaper than a dynamic one that is allowed to use the maximum number of nodes. Reducing the number of nodes to 8

[9] `https://calculator.aws/#/createCalculator/EC2`

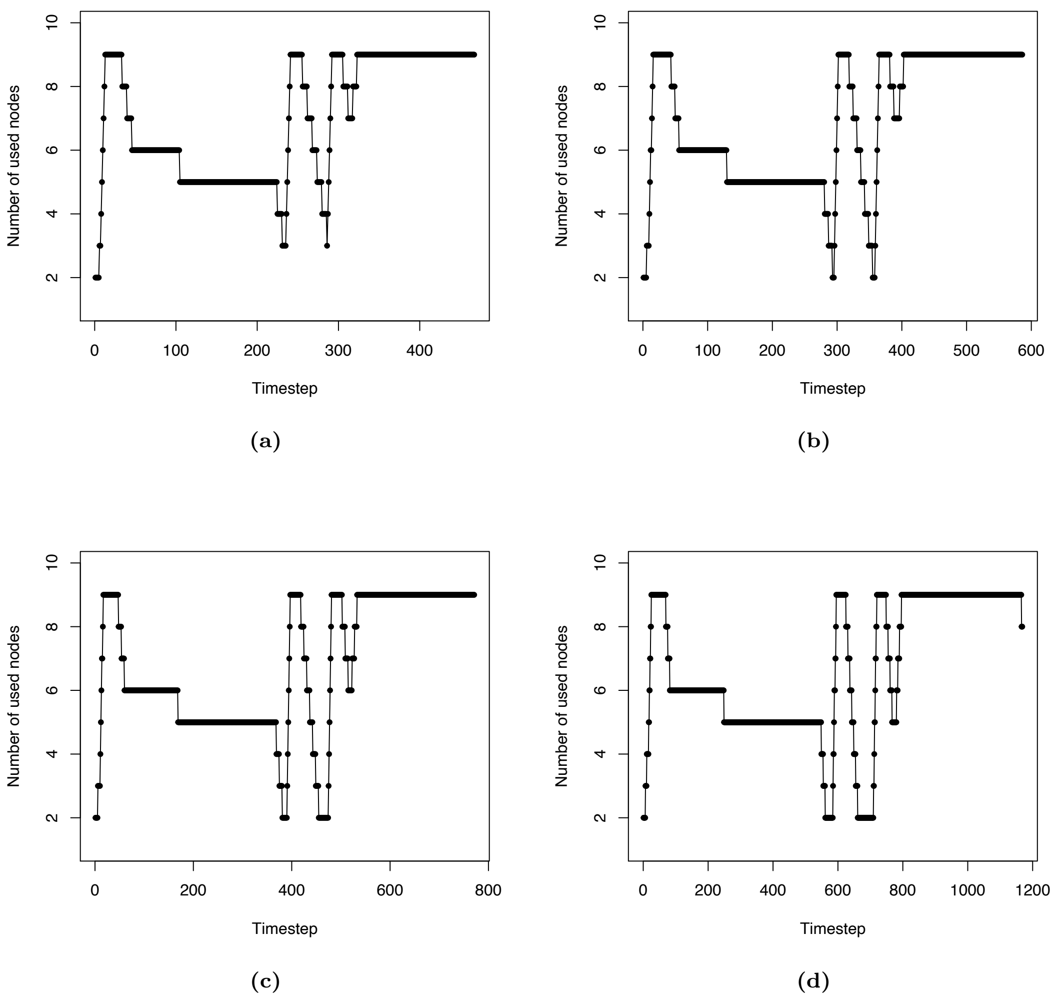
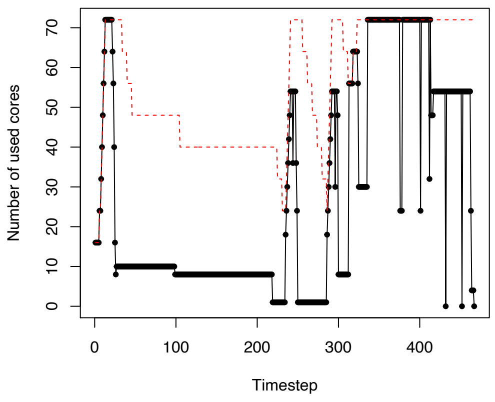
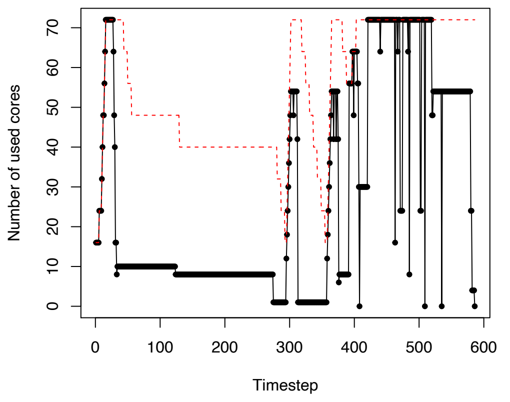


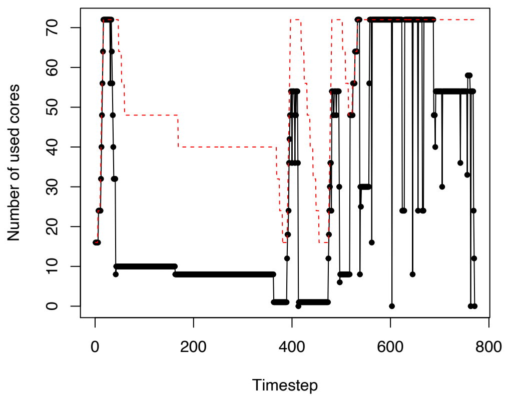
Figure 5.5: Figure illustrating the number of nodes available during the processed pipeline with regard to the different chosen VALET scheduler check intervals. The x-axes show the time step the data have been collected, the y-axes show the number of nodes available at a given check point. **(a)** 5 min check interval, **(b)** 4 min check interval (worst wall time run), **(c)** 3 min check interval (best, wall time run), **(d)** 2 min check interval.



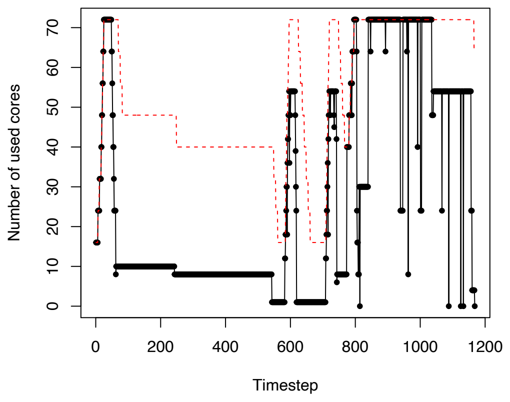
(a)



(b)



(c)



(d)

Figure 5.6: Figure illustrating the number of cores available (red, dotted line) and the current used cores recorded at this time step regarding the different chosen VALET scheduler check intervals (black). The y-axes show the number of cores available/used at a given check point, the x-axes show the time step where the data have been collected. **(a)** 5 min check interval, **(b)** 4 min check interval (worst wall time run), **(c)** 3 min check interval (best wall time run), **(d)** 2 min check interval.

leads to a wall time increase of around 4.5% compared to a full static cluster with the maximal number of nodes. For the 7 nodes scenario the overhead increased up to circa 11%. Further scenarios have not been tested as the wall time value would increase further and the dynamic clusters scenarios would perform better with respect to wall time and costs.

Finally, the configuration with the best performance has been taken, which was the one checking the load every 3 minutes and tried to tune it to stop nodes more aggressively if the load numbers are below the threshold. To achieve that, the weight W5 has been set to 0.6 instead of 0.2. Furthermore, the check time of the torque scheduler has been reduced to 5 seconds. The resulting values are marked with the entry "tuned" in the Check time column of Table 5.4. The collected values show a wall time overhead of 4.1% with around 24% of cost savings compared to a static cluster. Further explanations can be found in Section 5.4.

5.4 Discussion and Analysis

Before analyzing the collected results of the scheduler evaluation presented above, some lessons learned during the first test cycles are elaborated below. Throughout the first results, huge overhead values concerning the runtime have been measured. From the examined literature, the occurrence of an overhead could be expected, as the initial cluster has been taken as a starting point using two compute nodes, but not in the range of 20 to 30 percent. Therefore, the reason for these large overheads has to be found. The investigations pointed out that the used batch system scheduler's interval to check for new available resources, jobs can be scheduled on, was set to 600 second. That means that new nodes can be added to the cluster but will be firstly used after 10 minutes in the worst case, which led to a delay in the resource utilization and the increased wall times. After adjusting the lookup interval to 60 seconds, the overhead has been reduced to the reported numbers above. In order to avoid any race conditions leading to unwanted behaviors, the interval has not been reduced any further, except for the tuned run trying to push the dynamic scaling of the cluster to the edge. Another behavior that occurred during the first tests with low time values for the load check interval of the VALET scheduler was the use of more compute nodes than defined as maximum. The reason was a race condition between the reported state of the scheduler and the client executing the commands to add new nodes to the cluster. Sometimes, the nodes were deployed, but not recognized by the resource manager, so the maximal number of nodes was not reached already and the scheduler asked to deploy one more node. In the end, it can happen that one more node than the specified maximum has been deployed. This is not a problem if enough resources are available, but usually one takes all resources that are available and do not leave some spare ones. The problem has been solved by adding an additional cluster state (blocked) to indicate that a new node is in deployment and it needs to be waited until the next one can be added. These experiences can have a huge impact on the performance and also on the maximum resources that should be used. Another point to consider is setting an appropriate value for the parameter that specifies how long finished jobs reside in the queue history. Since the VALET scheduler uses information from already finished jobs to calculate resource estimates, the parameter should correspond with the expected job runtimes. The default value is rather low with 300 seconds, and needs to be adjusted. In order to have a larger historic data basis the value is set to a time period of 36000 seconds (10 hours).

The presented values were already collected with the improved lookup parameter of 60 seconds and the prevention of the described race condition. In general, the dynamically managed cluster using the implemented scheduler of this work performed well, the resulted overhead is in a range of around 4 percent, for the check interval of 3 minutes even lower. In total, this can be considered as a rather low value, especially compared to the number of resources that have been saved. Furthermore, the standard deviation of the wall time for each of the five conducted runs have been calculated to see how large the fluctuations between the different runs are. The results show that the deviations are increasing with the higher values of the load check interval. For 2 minutes, the standard deviation is rather low compared to the mean wall time with 2336 minutes. For the other chosen values, the standard deviation increases to around 9 minutes, still a very low value. The increase of the fluctuation can be explained through the rougher time resolution as the steps between the measurements are getting larger and therefore shortly occurring events might be missed or only detected sometimes. In summary, the results of the standard deviation calculation show that the results are reproducible for the implemented pipeline and thus allow a reliable estimation of the runtimes.

In general, the influence of the interval the implemented scheduler uses to check the load can be expressed as follows. The larger the interval is the slower the resources can be adapted. This leads to a slow increase of resources in a starting phase where the cluster queue is quite full. But it also leads to a slower down scaling of compute nodes if resources are unused and therefore more resources are available and can process larger amounts of future jobs faster. Of course the number of saved resources will not be as high if lower values are used for the check interval. Smaller test intervals in turn lead to a faster adjustment of resources and thus to greater savings in resources and costs. Due to the quicker down-scaling, there is no large capacity left to process larger job numbers directly if occurring suddenly. With regard to the test scenario used, the difference between the best mean wall time value (3 min interval) and the worst value (2 min interval) is about 0.5%. However, the difference in saved resources and costs is about 1% in favor of the smaller interval. So there is no real difference for intervals that are close to each other. But for larger intervals compared to the smaller ones the difference can increase. The difference in terms of resource savings between a 2 min and 4 min interval is about 3%. For larger intervals, the savings will become smaller and will to a certain degree approach a static cluster with maximum resources. In order to be cost efficient, low numbers for the check interval are recommended. Of course that can be dependent on the cost model of the used cloud provider. If the process of starting and stopping instances is priced in addition to the costs of running instances, larger intervals can be preferable.

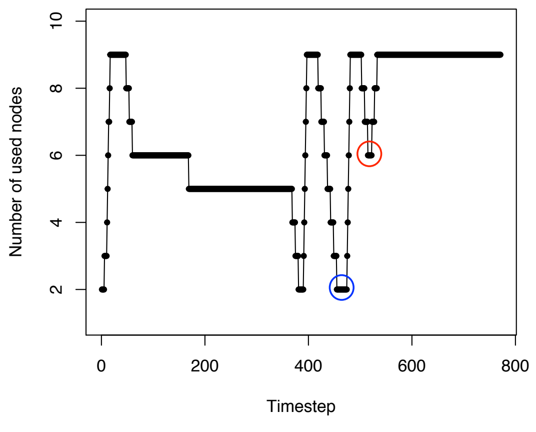
In addition to the savings on the cost side, the value of the average number of used compute cores indicates how many of the available cores are actually used over the entire period of the pipeline. For the static test cluster, a number of around 29 cores has been obtained whereas 72 were available the whole time. For all used interval variants the numbers were lower ($\sim$3%) which explains the longer wall times of dynamically scaled clusters. But this low overhead value shows that the scaling works satisfactorily as resources have been saved but the number of cores used is close to the ones of a static cluster. In order to get a better overview of the available cores and the cores currently used, these values have been plotted in the same graph (see. Figure 5.6) for each of the chosen VALET intervals to compare them with each other and get a better understanding on the impact of the different intervals. For all plots (a to d) the black curves are very similar as the pipeline has been implemented without dynamic parts to make constant measurements and to be able to compare the different interval values in the end. What can be deduced from the graphs, however, is that with larger intervals (e.g. 5 minutes) the curves do not slope as steeply as with smaller intervals and thus resources are scaled up and down more slowly and correspondingly fewer resources can be saved. Furthermore, it becomes visible how the available resources adapt to the actually used resources through the implemented scheduler. It can be seen in the first half of the graph that the difference between the used resources and the available ones is quite high for all intervals. This behavior is due to the more conservative chosen strategy, keeping more available resources even if the load is not that high to react faster on high peak demands. Especially at the end it can be seen that the lower the interval values the better is the resolution as more values are recorded in smaller steps which helps to reveal the behavior of the pipeline to find constant parts or parts with a high fluctuation in core usage.

Furthermore, smaller check intervals of the VALET scheduler lead to resource savings but also to a faster up-scaling of a cluster. The reason for this is the shorter interval and the resulting faster achievement of the weight-controlled scaling mechanism. The different weights triggered by different events through the scheduler are summed up over time to scale up a cluster even if no direct up scaling events are triggered. Especially in the phase where the initial cluster resources and no data about already finished jobs are available, it is possible to extend the resources of the cluster through the implemented weight-based mechanism. It is known that for example a large number of relatively long jobs are started at the beginning of a pipeline, it is advantageous to use a smaller interval in order to achieve a faster scaling of the cluster. Further, the implemented test pipeline and the data obtained from it regarding the scheduler behavior suggest that specific pipelines and workflows benefit from scalability when they include longer periods of low resource utilization in which a reduction of resources can be achieved.

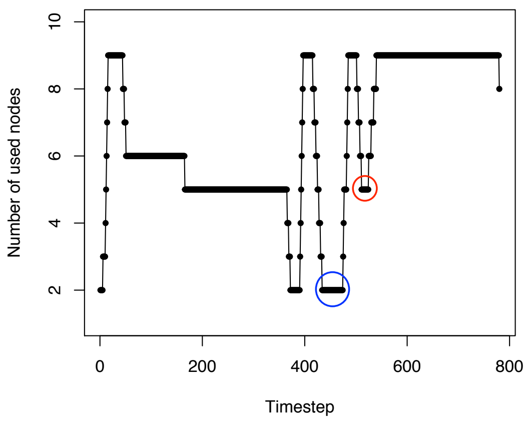
Finally, an attempt was made to further optimize the VALET scheduler parameters with respect to the existing test pipeline in terms of wall time and resource savings. Therefore results of a tuned variant using the best found configuration with a scheduler interval of 3 minutes were collected. The tuned variant showed a slightly higher wall time (0.5%) compared to the un-tuned variant. However, the parameter adjustment reduced the costs by 1.1%. A graphical comparison is illustrated by Figure 5.7. The graphical comparison shows the effect of the tuned parameters leading towards a more aggressive down scaling. This means that lower numbers of available nodes are reached faster and the time periods last longer, which can be seen at the steep down scaling to the initial cluster after time step 400. For the tuned variant (5.7b) the initial number of 2 nodes is reached earlier but also lasts longer. In summary, these stages lead to the lower costs and the resource savings. However, the tuned parameter lead also to the behavior that the wall time increases as resources are scaled down faster and need more time to be scaled up. This behavior is also visible in Figure 5.7. At the time steps of around 500, the last down-scaling peak, the tuned cluster goes down to 5 nodes where the un-tuned cluster is using 6 nodes. As the last step makes use of the whole available resources, the maximal resources are available faster for the un-tuned cluster and the wall time can be kept shorter than for the tuned cluster. Still, the differences are quit small within a range of around 1% and the question here is whether cost savings are more important than the runtime or vice versa.

5.5 Conclusions and Future Work

The goal of this work is to use different technologies to ease the process of deploying a virtual cluster environment using cloud resources. At the current development level VALET offers the possibility to deploy a virtual cluster on demand on OpenStack based clouds. It makes use of the flexibility of the virtualization technologies to resize a cluster dynamically according to its measured load. The deployment process is currently specific to an OpenStack environment mostly targeting public academic clouds or private companies operating their own OpenStack infrastructure. However, the implementation of the deployment process using Terraform offers the possibility to replace the OpenStack plugin with any other desired one. The whole configuration process is not affected by it. For the future a version for Amazon Web Services (AWS) and Microsoft Azure and also Google Cloud Platform would be valuable. This



(a)



(b)

Figure 5.7: Figure illustrating the number of nodes available comparing the un-tuned and tuned variant using a 3 minute interval for the VALET scheduler. The y-axes show the number of nodes at a given check point, the x-axes show the time step where the data have been collected. The related differences are highlighted in blue and red. **(a)** Un-tuned, **(b)** Tuned.

work showed that VALET is able to spawn an initial virtual cluster in less than 10 minutes, including one master node and two compute nodes as well as the complete configuration of the shared file system, resource management system, UNICORE middleware and the Zabbix monitoring system. Of course these values are dependent on different properties like the API speed of the used cloud system, the size of the used volumes and the available download/upload rate for the predefined images. As shown by the implemented test pipeline, the developed cluster resize mechanism works well and shows only a small overhead in comparison to a full static cluster, so it is able to save resources and thus costs. Furthermore, it was shown that the resize algorithm can adapt the number of nodes quite fast so that resources are not idle for longer time periods. The suggested default threshold values are rather conservative and can be further adjusted depending on the respective use case. It should be noted that the specific behavior depends on the respective pipeline, used tools and thus the job composition. Due to the influence of the specific pipeline, savings in resources and costs can vary accordingly. Nevertheless, the dynamic scaling of resources should be considered by users in advance of the data analysis in any case. Also beneficial for the future would be a collection of other workload scenarios that are publicly available to ease the choice of the threshold parameters and further ease the process of adapting them. In addition, through the integration of other resource management systems like Slurm or SGE (Son of Grid Engine) it would be possible to address a broader community.

Chapter 6

Security

6.1 Motivation

Security in the context of cloud computing and especially together with sensitive data is becoming increasingly important. For example in the field of personalized medicine, more sensitive patient data are being collected for subsequent analyses and evaluations. Since IT infrastructures of hospitals and research institutes can no longer cope with the increasing amounts of data, outsourcing the analyses to a cloud environment has become a cost-effective option. In order to adequately protect sensitive data and support the research in the area of personal medicine and other related areas, special cloud computing analysis environments with associated security concepts need to be available. In addition to the availability and use of special environments for sensitive data, a relationship of trust has to be established between a cloud user and a cloud provider. Trust can be built up through ongoing collaborations over a longer period of time. However, this is only occasionally the case for cloud users and providers. Another possibility is to proof one's trustworthiness with certifications granted by independent third party organizations. Granted certifications serve as evidence that an organization is compliant to a certain standard, for example with regard to IT security. Standards are issued by various organizations, for example from the International Organization for Standardization (ISO). In industry, it is a common concept to achieve relevant certifications in order to establish trust with customers. In academia, publications about certification processes are rarely found, especially in the IT sector, which has already been shown by the conducted literature research. However, the academic sector is also obliged to comply with the relevant legal requirements for handling sensitive data. Otherwise, future projects with reference to sensitive data can only be carried out difficultly or not at all. A certification in the area of IT security should therefore be of interest in the academic sector. The importance of a certification in the context of sensitive data is also supported by the survey results in Section 3.1 and Table 3.4 of this thesis. To help other organizations to handle and process sensitive data in a public cloud, concepts for secured analysis environments are presented below. Furthermore, the applied measures are evaluated regarding their risk reduction capability using known attack vectors. Another aim of this chapter is to provide insights into a certification process of ISO 27001 in an academic environment, from the selection of a suitable standard to the final audit.

6.2 Secured Compute Environment

The creation of secured cloud environments involves measures that have to be implemented on user side and on cloud provider side. In addition, a distinction has to be made between the analysis environments (VMs), as well as the sensitive data and its transfer and storage. In general, certain rules and conditions should be defined before approving a project that processes sensitive data in a cloud environment. These can include the following points: Who is responsible for the project? Who is part of the project and has permission to access the data to be processed? Additionally, who is the data controller and what type of data is available (e.g., image data, genomics data)? Furthermore, procedures should be defined on how to handle the data during its life cycle. Such standard operation procedures (SOP) should be recorded in a shared document and updated as needed. After the SOP document is set up and accepted from the cloud users and the cloud provider, the creation of a secured analysis environment can be started. An exemplary illustration of the suggested setup is shown in Figure 6.1. The setup presented here was developed for an OpenStack cloud using flat networks as well as a Quobyte storage system [42]. The center of a secured environment is the gateway VM (jumphost), which is used as a single entry point providing the only public access to the environment. Further, the jumphost is the connection between the public network and the VMs used for computational tasks. This setup is realized using different interfaces with separated networks. Thus, the jumphost has at least two different interfaces with two different IP addresses, belonging to separate networks. The first one is used for public connections, the second, internal interface establishes the connection to the compute VMs and therefore needs to be available on them as well. In addition to the internal network, compute VMs require another private network interface that is responsible for the connection to the storage unit. In total, three different networks are used within a minimal setup, where the compute VMs are only accessible via the jumphost. To further increase the security, the used networks are only accessible from specific projects and can be separated into their own VLANs (Virtual Local Area Network) to protect the network traffic. Depending on the setup, the network configuration can be created and managed by the user or by the provider. However, one task that falls within the scope of the respective users is the administration of the jumphost. The jumphost can be used to create local user accounts with different permissions. This involves the handling of SSH keys that need to be placed on the jumphost. This also means that the administration and thus the responsibility of the administrator account is on the user side. Provider usually have no direct access to user VMs. Users should also ensure that the firewall configurations of the individual VMs are as tight as possible, especially for the publicly available gateway VM. In order to work with sensitive data on the created cloud environment, they have to be transferred to and stored in it. Therefore, secure data transfer and secure data storage concepts have to be taken into account. Before a data transfer can start, it should be ensured that both endpoints are sufficiently secured. In addition, it should be checked whether the data at the starting point are in the expected state. In order to be able to perform a check after the transfer, a secure hashing algorithm (SHA2, SHA3 [218]) should be used to generate checksums of the data before transfer. Depending on the data size and the available transfer speed between the endpoints, different transfer protocols can be used. In the simplest case, the applications scp [219] and rsync [220] can be used for data transfer. In the field of bioinformatics, the Fast Adaptive and Secure Protocol developed by IBM and the associated application Aspera are becoming increasingly popular, as it promises secure and high-throughput data transfers [221]. For all applications used, sufficient encryption of the data traffic needs to be ensured, where an asymmetric encryption mechanism might be favorable to expose as less information as possible to the provider. To further secure a data transfer, the endpoint can be a standalone VM within the cloud project, that has no direct relation to the analysis environment. After the transfer is finished, these VMs could be deleted completely. It can also be considered to allow direct access to the storage unit from a transfer VM instead of storing data locally and push it to the storage unit after the public access has been disconnected. Furthermore, VMs designed purely for the purpose of data transfer can be hardened and restricted by firewall rules, like access restrictions to single IP addresses, to reduce the risk of an attack. After a data transfer is completed, the checksums can be compared with the initial checksums to detect any changes. In addition to the general data access, which is only possible from internal networks, the read and write permissions should be adjusted with regard to the authorized users. The functionality of fine granular access rights should be supported by the storage solution used. In order to assign permissions on a per user base, the built-in function of the Quobyte storage system for the creation of X.509 certificates has been used. The created certificates can be linked to the desired volumes, which regulates both, access and the corresponding permissions. Since the creation of the certificates and the granting of rights in this setup is the responsibility of the cloud provider, an additional, specially secured VM was provided, which holds the correspondingly required user certificates. Selected users are able to access the VM via SSH to get their individual certificate. The existence of concepts for the creation of secured environments is a first step to handle sensitive data. However, it is difficult to verify whether such a concept delivers what it promises. Likewise, such a concept provides little information about the general IT security of a cloud provider. In order to demonstrate appropriate care in the area of IT security and thus build trust with users, successful certifications are necessary. An overview of available certificates in the area of IT security and their properties can be found in Section 6.3.

6.3 Certifications

Certifications are a possibility to demonstrate, that applied measures and regulations are compliant with a certain standard. In the IT security and cloud context many standards are available [144, 145]. For a cloud provider, the question arises which one fits best regarding the implementation and which one has the greatest public perception. At the beginning of 2019, a certification process in the area of IT security was initiated in the context of the de.NBI Cloud in order to create trust with regard to the processing of sensitive data. The goal is to certify all participating cloud sites according to the same standard. From the available standards, ISO 27001, BSI C5 and CSA STAR were assessed as promising and subject of a detailed evaluation. A detailed description of each standard is presented in the following sections.

6.3.1 ISO 27001

The ISO/IEC 27001 standard is provided by the ISO organization. It describes best practices to implement an ISMS and follows a risk based approach. The current version was released in 2013 an update is planned for 2022. In addition to ISO 27001, the ISO 27002 is provided, which can be used as a guide for implementing the measures described in ISO 27001. The norm is divided into seven initial chapters (requirements) and 14 further chapters, which correspond to the Annex A of ISO 27002. These main chapters contain further sub chapters with references to the respective topic of the corresponding chapter. In total, Appendix A contains 114 controls that have to be taken into account accordingly. The topics covered in the individual chapters range from general information security guidelines to personnel security, asset management, cryptography, supplier relations, compliance aspects and more. Finally, the guidelines and documents to be prepared have to be supplemented by the Statement of Applicability (SoA), which needs to include information of the applicability for each individual control, as well as further explanations of the reason for their inclusion and the status of their implementation. The SoA can be prepared, for example, in text form or in tabular form. At the core of the standard is the recording, assessment and treatment of risks. A risk analysis should refer to the organization's assets, these can be physical, like servers or switches, but also include the knowledge of

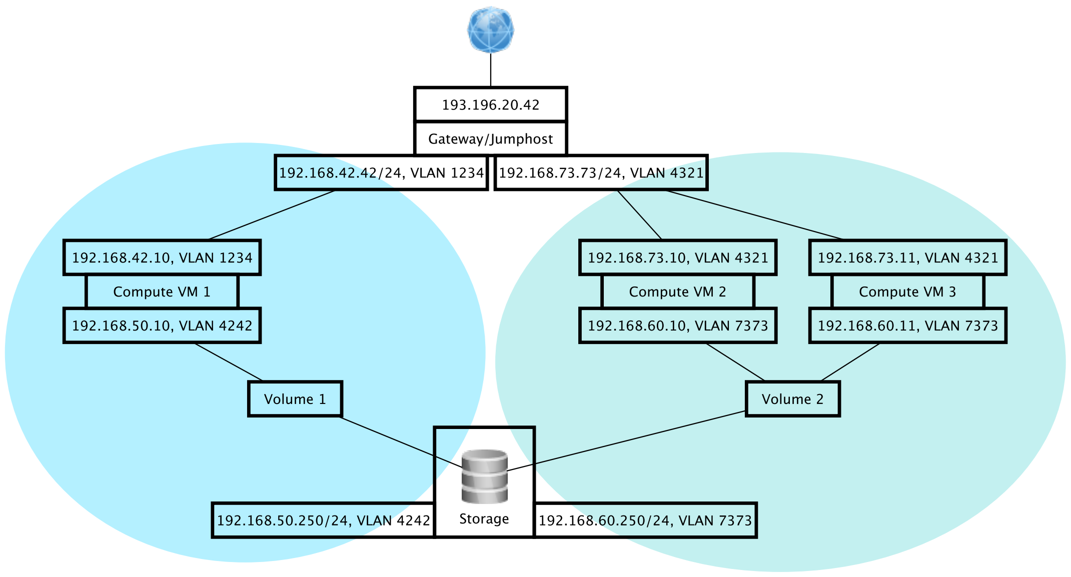


Figure 6.1: Exemplary use case illustration of the secured setup, including arbitrary IP addresses and VLANs, to clarify the nesting of the different used network interfaces and their function. In this use case a single jumphost is used to provide access to three Compute VMs and two storage volumes, whereby the data and their flow of VM 1 (left bubble) are separated from VM2 and VM3 (right bubble) and also from the corresponding volumes. The separation is achieved through the use of VLANs and illustrated by the two bubbles. The VLANs 1234 and 4321 are used for the internal connection from the jumphost to the Compute VMs as well as from the Compute VMs to the storage unit (4242, 7373). Further, this example shows two possible scenarios, an isolated analysis scenario (left) where only a single VM is able to access the data and a shared analysis scenario (right) where two VMs have access to the same data. This example shows a setup similar to a real project, where the raw data are kept isolated and accessible by very few people but the processed data are made available to multiple users for further analysis steps. Furthermore, this example setup could be extended by multiple bubbles, volumes or VMs.

employees or the content of documents. If all relevant controls are documented and measures are implemented, an authorized auditor can be commissioned via a certification authorization organization to review the existing documents and the implementation status to finally issue a recommendation for or against the granting of a certification. In case of an initial ISO 27001 certification two phases have to be passed. In phase one, all documents are checked for presence by the auditor. If phase one is successfully passed, phase two follows. During a several days lasting audit, the relevant premises are inspected and random samples of the individual controls are carried out, which serve as evidence for their implementation. After a successful initial certification, annual repeat audits have to be carried out to track the progress of the implemented ISMS.

6.3.2 BSI C5

BSI C5 is a cloud-specific norm developed by the German Federal Office for Information Security that describes a set of minimal requirements that should be fulfilled by cloud providers. The current version was released in 2020[10]. The criteria developed were based on other national and international standards: ISO 27001, ISO 27002, ISO 27017, BSI IT-Grundschutz-Kompendium, CSA Cloud Control Matrix in version 3.0.1, AICPA (American Institute of Certified Public Accountants) - Trust Services Criteria 2017 (TSC) and the ANSSI (National Cybersecurity Agency of France). The criteria catalog contains 17 sections covering different topics like general information about the offered cloud service, organization of information security, human resources, identity and access management, security incident management and many more. In total, 121 controls are listed in the C5 catalog. In addition to the basic requirements, extended requirements are described that can be additionally fulfilled in order to raise the safety level. Once the documentation is completed and all measures can be checked for effectiveness, a qualified auditor can be appointed. A qualified auditor under the C5 catalog must have at least 3 years of professional experience in the context of IT audits or hold a personal certification for example as an ISO/IEC 27001 Lead Auditor. As proof of a passed audit, a corresponding attestation is awarded.

6.3.3 CSA STAR

The CSA STAR is another cloud-specific standard that is widely used and that builds upon other standards and frameworks like Service Organization Control 2 (SOC 2) related to criteria from the AICPA (Trust Service Principles) and ISO 27001, combined with own developed criteria. Furthermore, the STAR

[10] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/CloudComputing/Anforderungskatalog/2020/C5_2020.pdf?__blob=publicationFile&v=2

standard includes a maturity model that determines the maturity level of the implemented measures on a scale of 1 to 15, where 15 is the highest value (innovative). For STAR, CSA provides the CCM (Cloud Controls Matrix) and the CAIQ (Consensus Assessments Initiative Questionnaire) in its latest version 4.0.2 at the time of this theses. The CCM includes 17 topics with a total number of 197 controls. The domains cover topics such as Change Control and Configuration Management, Infrastructure and Virtualization, Mobile Security or Threat and Vulnerability Management. In addition, the CCM contains a column that specifies the responsible party for each control distinguishing between IaaS, PaaS, and SaaS. In addition to the CCM, the CAIQ is available which is a list of questions related to the individual controls of the CCM. On the one hand the questionnaire catalog is used to enable a customer to obtain information about a cloud provider. On the other hand it is used to enable a cloud provider to determine the status of its own measures more easily. In total, the CSA STAR standard offers three possible levels of certification or attestation. The first is a self-assessment, performed by a cloud provider itself. Level 2 includes an examination by a third party auditor. Level 3 is aimed at critical facilities and requires a continuous, automated monitoring and adaptation of the implemented measures. In addition to the CCM and the CAIQ, CSA maintains a publicly accessible registry[11] where certified cloud providers can register themselves with their appropriate security level, for the sake of transparency towards potential customers.

In Section 6.4.2, the standards presented are compared to each other regarding their compatibility. Furthermore, experiences with an ISO 27001 certification process within an academic environment and a small team are described.

6.4 Results

The first part of this section reports the results regarding the analysis of the secured environment and its impact on general cloud security threats and data specific threats. As mentioned above, the second part presents the compatibility comparison results of the selected IT security standards.

6.4.1 Secured Compute Environment

The concept for a secured environment to handle sensitive data is evaluated below using known threats. First, the concept has been evaluated against general, but cloud-specific threat scenarios including the Egregious Eleven published by the CSA[12] and two further scenarios from the Treacherous Twelve, also published

[11] `https://cloudsecurityalliance.org/star/registry/`

[12] `https://cloudsecurityalliance.org/artifacts/top-threats-egregious-11-deep-dive/`

Threat	Responsibility	Impact
Data Breach	Shared (User)	Strongly reduced
Misconfiguration and Inadequate Change Control	Shared	Reduced
Lack of Cloud Security Architecture and Strategy	Shared (Provider)	Reduced
Insufficient Identity, Credential, Access and Key Management	Shared (User)	Slightly reduced
Account Hijacking	Shared (Provider)	None
Insider Threat	Provider	Reduced
Insecure Interfaces and APIs	Shared	None
Weak Control Plane	Shared (Provider)	None
Metastructure and Applistructure Failures	Provider	None
Limited Cloud Usage Visibility	Shared	Reduced
Abuse and Nefarious Use of Cloud Services	Shared	Reduced
Denial of Service	Provider	None
Shared Technology Issues	Shared (Provider)	Strongly reduced

Table 6.1: Impact evaluation of the secured environment concept with reference to the Egregious Eleven as well as the two last threats which were taken from the Treacherous Twelve. The Impact column indicates whether the presented setup is able to reduce the risk of the corresponding threat and if so, how much: slightly reduced, reduced, strongly reduced or none (no impact at all). Furthermore, it is shown whether a threat is the responsibility of the service provider, the user or both (shared), with more portions to the party in brackets.

by CSA[13]. Second, the concept has been examined specifically to data security issues. For both, the work of Choudhary et al. [222] has been used for the specified threat scenarios and their evaluation. An overview of the used scenarios as well as the impact of the safety concept can be found in Table 6.1 and 6.2. Detailed explanations of the different threats and issues and the impact of the presented setup are given below.

- Data Breach: The risk of a data breach has been strongly reduced with the presented setup. The number of entry points were reduced to a minimum, using a single jumphost machine. Further, access to the jumphost is granted via SSH keys, that have to be placed explicitly by a project manager owning root access to the jumphost VM. The restricted public access is complemented by personalized X.509 certificates to access volumes containing sensitive data. Furthermore, fine granular access rights such as read and write can be granted via certificates. The responsibility is shared between provider and users, but with a larger portion for the users as the users are in charge of the analysis environment including the placement of SSH keys and corresponding network access rules.

- Misconfiguration and Inadequate Change Control: The configuration of the analysis environment is mostly done by the users the project belongs to, but some work might be done by a provider depending on the setup.

[13] `https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdf`

In this case and environment, the cloud provider is responsible for the correct network configurations including VLANs and IP address ranges but also for the creation of the certificate secured volumes of the Quobyte storage system and the handling of the certificates itself. Therefore, the responsibilities in terms of misconfigurations can be seen as shared. The risk of a misconfiguration is reduced as a provider should have the knowledge to avoid mistakes. Furthermore, a provider can recommend configurations and help users to avoid misconfigurations. These measures reduce this specific risk but it can not be strongly reduced or eliminated.

- Lack of Cloud Security Architecture and Strategy: In order to increase the security level of a cloud environment and virtual environments like the presented one, it should be directly designed for this. To develop and implement a security concept, a deeper knowledge of cloud computing and IT infrastructures is required. Not all cloud users have this kind of knowledge, therefore planning and implementation should be carried out by employees of a cloud provider as they have the relevant knowledge. In this work, the conceptualization was carried out by people familiar with cloud technologies and also acting as a cloud provider. Therefore, the presented environment was created from the very beginning with regard to security to reduce the risk of this threat. Due to the architecture, the cloud provider is mostly responsible for this threat, but the project users are still in charge as they have to configure the environment regarding their own needs.

- Insufficient Identity, Credential, Access and Key Management: The result of a successful attack due to an insufficient management of credentials can have severe consequences and impacts. For example, attackers can get access to areas they should not have permissions to or can even change between accounts and misuse them. In general, a cloud provider is responsible for handling any kind of credentials with necessary care, meaning that used technologies and access permissions are in place and are working correctly. For the de.NBI Cloud, a single sign on (SSO) mechanism via ELIXIR AAI [64] is used. This means that during the login process users are forwarded to their corresponding identity provider, their institution for example, and have to sign in via their credentials. The use of a SSO procedure reduces the cloud provider's risk because only a pseudonymized 40 characters long identifier is stored on the cloud infrastructure. Therefore, no passwords or other information can be stolen. The management for providing the personalized certificates from the Quobyte storage is done via secured VMs running in a separate project managed by employees of the cloud provider. User requiring a certificate need to login to a specified VM with a locally created user via SSH. Furthermore, the VMs providing the certificates will be shut down if not actively needed. All these measures are done from provider side to manage as few credentials and keys as possible and to protect them accordingly. From user side the management of the SSH keys on the jumphost VM has to be done on its own, a provider can help providing best practices but the users are still responsible. Likewise, the SOP document created at the beginning helps both sides to keep track of who is allowed to access certain data and to what extent. In summary, the measures taken reduce the threat cited, but only to a lesser extent since the user side cannot be controlled by the cloud provider nor should it be.

- Account Hijacking: Privileged accounts are interesting targets for attackers as they have a wide range of permissions and thus enable further attack scenarios which is why they have a great potential for damage. As only cloud providers should own privileged accounts, the responsibility lies strongly on their side. In the context of the working environment presented here, the account for managing the jumphost (user responsibility) can be considered as a privileged account. This kind of accounts should only exist once, which at least minimizes the success of phishing attacks; otherwise, the risk cannot be further reduced by the measures taken.

- Insider Threat: Insider attacks can happen due to hijacked administrator accounts, insecurely stored credentials, misconfigurations or criminal insiders. The risk of an insider attack has been reduced by means of this concept in the form that employees of the cloud provider do not initially have access to the individual VMs of the created environment. Also, employees do not have permissions to access any certificate secured volumes by default. However, an attacker can give himself access depending on the scope of the obtained permissions, but this requires that the processes and methods for this are known. Furthermore, the employees were carefully selected, and this should be taken into account as much as possible during the hiring process. Of course, the risk of an insider attack at infrastructure level usually lies entirely with the cloud provider. Within the working environment, the project manager is responsible for the users and their data access. An insider attack can also be a risk here, but this cannot be prevented through the environment directly. This needs to be limited by further configurations of the environment, such as the regulation of data traffic in external networks, which should be done within the cloud project and not on infrastructure level.

- Insecure Interfaces and APIs: Since no additional APIs (Application Programming Interfaces) or other interfaces have been created with the setup of the virtual environment, it has no negative or positive impact on the risk reduction of this threat. APIs or interfaces used or implemented by users are not included here but of course these can be a security risk too and should be well-considered.

- Weak Control Plane: As no control plane is involved in this setup, there is also no impact on this threat.

- Metastructure and Applistructure Failures: Since the virtual environment created is not different to other projects or the used resources, it has no effect on the risk of this threat.

- Limited Cloud Usage Visibility: The title of this threat may be unclear at the beginning. What is meant is the release of suitable applications in a cloud environment by an organization. If approval policies are circumvented or prohibited applications are used without approval, which can lead to security risks. This is also referred to as shadow IT. Furthermore, it is difficult to prevent or detect intrusions of attackers if the application used for this purpose should not be available. The scenario described does not quite apply here, because the presented cloud environment is a research environment in which the users do not use specific applications or services alone. Users are able to install any desired software within VMs or implement it themselves. Likewise, the applications used are not approved by any organization. However, the attack surface is significantly reduced through the use of the jumphost. This restricts the accessibility of VMs for data analysis purposes, which makes it more difficult to exploit application-specific security vulnerabilities. In general, the responsibility of this threat is shared between provider and users depending on the granted permissions. In the present scenario, the responsibility lies more with the user or project manager because they have access to the individual VMs.

- Abuse and Nefarious Use of Cloud Services: The possibility of cloud resources being used for malicious purposes is still present with this setup. The presented concept has no influence on the infrastructure itself. However, the virtual research environment can of course be misused, for example to distribute malware or initiate DoS attacks. However, the risk has been reduced as there is only one publicly accessible entry point with the jumphost which can also be further secured according to the individual users. The responsibility for this risk lies with the provider at the infrastructure level, but users are in charge with regard to VMs. This means that the responsibility can be seen as shared between provider and user.

- Denial of Service: Service interruptions due to intentional traffic overload can be prevented at infrastructure level by redirecting and distributing traffic to different load balancers and regions. Apart from the infrastructure on which the virtual environment has no direct influence, the risk of a denial of service attack cannot be reduced with the initial concept presented here. In any case, a DoS attack my prevent data access but has no impact on data integrity or data loss. A possible solution is discussed and presented in Section 6.5.

- Shared Technology Issues: Since the Meltdown and Spectre security vulnerabilities became public [223], it became clear that a separation is not perfect even in the area of virtualization. The access and reading of unauthorized memory addresses is a threat that affects not only projects with sensitive data, but these in particular. This is highly relevant in the context of cloud computing as resource sharing is one of the primary concepts leading to cost efficiency. However, this also results in the risk of unauthorized access since the same resources are used by different users. However, this risk can be greatly reduced by the presented concept. One measure is the separation of resources and their assignment to separate projects. This applies in particular to project specific networks that are not accessible to any other projects. However, the network traffic is still routed through the same infrastructure as for any other project. In order to further encapsulate the data traffic separated VLANs are used to achieve a segregation at infrastructure level. In addition to the network traffic, and apart from the created environment itself, VMs can be placed on dedicated hypervisors to prevent unauthorized memory access. Data are still stored on the general storage system, but separated into volumes secured by X.509 certificates and a project specific access network. These measures significantly reduce the risk of unauthorized access through the use of shared resources in a cloud environment.

Issue	Responsibility	Impact
Segregation	Shared (Provider)	Strongly Reduced
Isolation	Shared (Provider)	Reduced
Location	Provider	None (Strongly Reduced)
Provenance	User	None
Remanence	Shared	Slightly reduced
Integrity	Shared (User)	Reduced
Lineage	User	Slightly reduced
Leakage	Shared	Reduced
Backup	User	None
Recovery	User	None

Table 6.2: Evaluation of data-specific issues and the impact of the presented secured analysis cloud environment. The impact is specified in the categories slightly reduced, reduced and strongly reduced or none if an issue is not affected by the presented setup. Furthermore, the responsibility of the different issues is assigned to the provider, the user or shared if both are involved. If the responsibilities are shared but not equally, the party in brackets is more responsible.

In addition to general threats in the cloud context, data specific problems can be identified. The impact of the security concept presented is evaluated in the following with regard to the selected issues.

- Segregation: A strong data segregation is achieved with this concept. First, data access is only possible through project specific networks. Second, network traffic of these specific networks is separated from other ones using VLANs. Third, within a cloud project, access to Quobyte volumes is secured using X.509 certificates. Via the personalized certificates, read and write permissions can be further individualized. Furthermore, it is possible to associate certificates with multiple volumes. This enables an additional data-separation-layer. A scenario in the area of sensitive genomic data would be the separation of raw data from pre-processed or already aggregated data. The responsibility can be considered as shared, with larger shares on provider side. On the present infrastructure of the de.NBI Cloud Tübingen the provider is responsible for the project specific networks, the corresponding VLANs and the certificates. On user side, the specific networks have to be used as described. Furthermore, access credentials, keys, and personalized certificates have to be kept private.

- Isolation: For data isolation, the same concepts apply as for data segregation, but it should be noted that stored data still resides on a shared storage unit. Data that resides in RAM can be protected from foreign access using dedicated hypervisors. This reduces the issue of data isolation. As for segregation, the responsibility can be classified as shared with larger shares on side of the provider.

- Location: With regard to the cloud infrastructure of the de.NBI Cloud site Tübingen used in this work, the location of stored data can be clearly determined. The data is located in Tübingen, Germany in one or both data centers as no outsourcing of data takes place. Therefore, the issue of data location can be seen as strongly reduced. Assumed that users have no explicit influence on the data storage location, the cloud provider is fully responsible for it.

- Provenance: Data provenance is not affected by the analysis environment, this topic has to be handled by the data owner or cloud users.

- Remanence: In general, the virtual environment has no amplifying or reducing impact on this problem. However, data remanence can slightly be reduced by the procedures recorded in the SOP. The SOP should also include agreements regarding further data handling such as deletion or archiving, when the end of a data lifecycle is reached. Through agreements with the provider, secure deletion processes can be initiated and executed accordingly. The responsibility can be seen as shared because users have to take care that data are deleted appropriately or to contact the provider and initiate the secure deletion process. The provider has to make sure that the required steps are processed correctly.

- Integrity: Data integrity is a highly relevant issue in the context of sensitive data as smallest differences can have a large impact on the obtained results. In general, the analysis environment has no direct effect on the data integrity itself. However, limited access during the analysis phase, where data have already been transferred to a cloud environment, helps to reduce data integrity issues. It also simplifies the tracing process in case of problems as the access is restricted to a specific group of people. During the transfer, the integrity of data can be verified using checksums created before and after the transfer. In general, the responsibility for data integrity lies with the data owner or users. However, the provider should ensure that the storage solution used does not lead to any integrity problems.

- Lineage: Data lineage is a relevant issue during the process of data analysis, especially in the context of analyses pipelines or artificial intelligence. Of course the obtained results are of interest, but it is necessary to understand an analysis process in order to be able to verify and reproduce obtained results. From the provider's point of view, the lineage of data is a task of those who analyze data and the virtual environment has no influence by default. Only in the first step, the data transfer, a provider can become active and confirm the arrival of the data and possibly its integrity.

- Leakage: A data leak is one of the worst scenarios, especially if sensitive data are involved. The presented concept for a secured environment reduces the risk due to different measures. First, the jumphost VM serves as single public access point. The compute VMs having access to sensitive data are not publicly accessible. Depending on the individual working principles of a project, it is possible to cut any incoming and outgoing connections from the compute VMs, except for the ones necessary. Leakage of data can happen on purpose or accidentally from both sides, provider or user. A provider can cause a leakage due to misconfigurations or malicious employees. The same applies to users. Therefore, the responsibility is noted as shared.

- Backup and Recovery: Backup and recovery concepts are important with regard to data that are difficult to replace, maybe due to their size or their availability. However, if not otherwise noted or offered as a service by a cloud provider, data backups and their recovery are up to users themselves. The presented environment does not include any backup or recovery concepts.

6.4.2 Certifications

In order to gain a better insight into the decision-making process for one of the aforementioned standards, a comparative mapping of controls at domain level has been performed. For this purpose, the mapping tables provided by BSI[14] and CSA[15] were used. In addition to the mapping, the available gap (CCM) and level (C5) analyses have been taken into account to find relevant differences between the standards. Controls marked with a minus sign in the C5 catalog indicate that the security level of the mapped C5 control is estimated higher than the compared one. For the sake of simplicity, it will be referred to as partially consistent with the CCM notation. The results and conclusions are not affected by this renaming. The comparison has been conducted deliberately on the higher-level domain categories of the individual controls to ensure clarity and to not go beyond the scope of this work. A tabular overview of the mappings can be found in Table 6.3 and Table 6.4. In order to make this section more readable only the domain abbreviations are used in case of C5 and CSA. The full names are listed in Table A.1 and Table A.2.

In summary, the CCM consists of 197 controls, where 77 (39.1%) could be mapped directly, 92 (46.7%) could be mapped partially, and 28 (14.2%) could not be mapped at all. Similarly, it can be seen from the scattered mapping of the individual ISO controls that the controls and domains of the CCM are structured differently. Controls that are summarized in a single chapter of the CCM are spread over many chapters in ISO 27001. Most of the annotations concerning the partial gap level describe missing specifications that are not stated explicitly in the ISO 27001, but in the CCM. One missing specification, the review period of individual controls is often noted, which in the case of CSA STAR has to be done at least annually, whereas for the ISO standard a review has to be done regularly, but not necessarily annually or at shorter intervals. Thus, the controls marked as partial gap can be considered compatible with each other for the most part. More interesting are the controls that could not be mapped as they can reveal more striking differences between the individual standards. The full gaps listed by CSA are related to the domains AIS, BCR, DCS, DSP, HRS, IAM, IPY, IVS, LOG, SEF, TVM and UEM. An analysis of the non-mappable controls is presented in Section 6.5.2.

The evaluation of the C5 controls mapping to ISO 27001 controls showed that from the total number of 121 controls in the C5 catalog, 37 (30.6%) could be mapped directly, 71 (58.7%) partially and 13 (10,7) not at all. The largest proportion is allotted to the partial category, which means that controls could be mapped, but the level of the C5 controls is considered higher by the BSI. The mapping of the domains between both catalogs is similar as no large scattering of the different controls has been observed. The first domain (OIS) matches with the first chapters of the ISO standard (4-10 and Annex 5, 6) and so on. Different domains and the controls contained are referenced multiple times, but not to the extent as for the CCM. The controls specified as a full gap are found

[14] `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/CloudComputing/Anforderungskatalog/2020/C5_2020_Referenztabelle.xlsx`

[15] `https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/`

CSA CCM	ISO 27001	Gap Level
A&A	9, A.18	Partial (3)
AIS	9, A.5, A.7, A.12, A.14, A.16, A.18	Partial (3), Full (1)
BCR	5, 6, 7, 8, A.5, A.7, A.12, A.16, A.17	Partial (7), Full (4)
CCC	A.5, A.12, A.14, A.15	Partial (3)
CEK	5, 8, 9, A.5, A.6, A.8 - A.10, A.12 - A.16, A.18	Partial (5)
DCS	A.8, A.11, A.17	Partial (4), Full (3)
DSP	5, A.5, A.8, A.11 - A.14, A.18	Partial (8), Full (5)
GRC	1, 4, 5, 7, A.5 - A.7, A.18	Partial (4)
HRS	A.6 - A.8, A.11, A.13	Partial (6), Full (1)
IAM	A.6, A.8, A.9, A.12	Partial (6), Full (3)
IPY	A.14, A.15, A.18,	Partial (1), Full (2)
IVS	5, 6, 7, 9, A.5, A.9, A.12 - A.14	Partial (9)
LOG	A.10 - A.12, A.16, A.18	Partial (2), Full (1)
SEF	4, A.6, A.16, A.18	Partial (3), Full (2)
STA	5 - 9, A.5, A.7, A.15	Partial (14)
TVM	5, A.5, A.6, A.7, A.9, A.10, A.12, A.13, A.15, A.16	Partial (4), Full (2)
UEM	A.6, A.8, A.9, A.11, A.12 - A.14, A.18	Partial (13), Full (1)

Table 6.3: Mapping of CSA STAR control domains to ISO 27001 control domains. Furthermore, a survey of gap levels was included. Partial refers to an incomplete mapping between a CCM control and an ISO 27001 control. Full means that a control from the CCM could not be mapped at all to a control from ISO 27001.

in the domains SP, PS, OPS, IDM, PI, SSO, INQ and PSS. The proportion of unmappable controls in the C5 catalog is smaller (10.7%) compared to the CCM (14.6%). This also applies for the relative number of equivalently mappable controls, which is 30.6% for C5 and 39.1% for CCM. Further considerations can be found in Section 6.5.2.

6.5 Discussion and Analysis

In the following, the respective results of the secured environment as well as the certification part are discussed and analyzed in more detail.

6.5.1 Secured Compute Environment

Various measures have been applied to increase the security of a virtual analysis environment to enable a secure processing of sensitive data. To restrict the access, one VM (jumphost) has been chosen as the only public access point. This reduces the area for attacks from the outside and puts an additional layer between sensitive data and the public. A disadvantage of this concept is that it creates a single point of failure. If the jumphost is unusable, access to the compute VMs is no longer possible. For example a DoS attack could use such a single point of failure as a target. To avoid service interruptions and failures, the setup could be extended by one or more additional jumphosts, which can serve as loadbalancers or backups in a high availability setup. This would compensate the

BSI C5	ISO 27001	Gap Level
OIS	4-10, A.5, A.6	Partial (4)
SP	A.5	Partial (2), Full (1)
HR	A.7, A.13	Partial (5)
AM	A.8	Partial (5)
PS	A.9, A.11, A.17	Partial (4), Full (2)
OPS	A.9, A.12, A.13, A.14, A.17, A.18	Partial (16), Full (6)
IDM	A.6, A.9, A.12	Partial (7), Full (1)
CRY	A.10, A.13, A.14, A.18	Partial (1)
COS	A.13, A.14	Partial (6)
PI	A.11	Partial (1), Full (1)
DEV	7, 8, A.7, A.9, A.12, A.14	Partial (5)
SSO	A.7, A.15	Partial (3), Full (2)
SIM	A.6, A.16	Partial (1)
BCM	A.17	Partial (3)
COM	9, A.12, A.18	Partial (1)
INQ	None	Full (4)
PSS	A.9, A.10, A.12, A.13, A.14, A.18	Partial (7), Full (5)

Table 6.4: Assignment of BSI C5 domains to ISO 27001 domains based on their controls. The gap level column indicates, whether controls could be mapped only partially (partial) or not at all (full). The numbers in parentheses show the number of occurences per domain.

failure of a single jumphost. In addition to a jumphost as a single entry point, a separation at network level has been achieved by using project specific networks and VLANs. Setting up networks from the provider's side has the advantage that users cannot change them independently and thus cannot misconfigure them. Furthermore, the provider usually has a greater expertise in setting up the networks correctly. A disadvantage of this workflow is the increased workload for the cloud provider. Another measure is the restricted data access at user level using personalized X.509 certificates. First, this enables a granular assignment of permissions without affecting other users and second, makes individual tracking of conspicuous activities possible. For the infrastructure of the de.NBI cloud site Tübingen, the assignment of permissions and the creation of certificates takes place in the storage backend on provider side. This ensures that the control remains with the provider who is usually more familiar with the technologies used than the common novice users. The direct control of the cloud provider enables it to act in emergencies such as compromised user accounts and accordingly stopping malicious attacks. Disadvantages are again the increased workload for the cloud provider and the raised risk of insider attacks due to misconfigurations or malicious intentions of employees.

Most of data specific issues have been reduced by the presented measures. Of special interest in the context of sensitive data should be the data location issue. In most cases, it is not clear to the user of a cloud environment where stored data are physically deposited. Depending on the country, different laws apply with regard to access and protection, which do not necessarily have to correspond to those of the country in which the data have been collected. Accordingly, when choosing a cloud provider, attention should be paid to the physical data location. Important issues, that could not be addressed or solved with the presented security concept, are backup and recovery. The reason for this is that backup and recovery strategies strongly depend on the available infrastructure and its capabilities. Therefore, it is difficult to directly implement a general backup and recovery strategy within the setup. Another problem is the distribution of the data by additional copies, this should be communicated to the data owner in order to guarantee a correct handling regarding storage location and deletion processes. This kind of procedure should also be recorded in the associated SOP document.

A final remark should be made regarding the usage of SSO procedures. SSO technologies are convenient for users and providers. Users can simply use their usual credentials and do not have to create an additional account. For providers it offers the advantage that only a user name has to be stored, which can be independent of the actual credentials of the associated identity provider. In case of the ELIXIR AAI (Authentication and Authorization Infrastructure), this is a unique 48 digits long string consisting of letters and numbers. A risk of SSO mechanisms that should not be neglected is the compromise of such accounts. By taking over a single account, an attacker can gain access to additional services connected to this account.

6.5.2 Certification

The conducted mapping of the two cloud-specific standards with regard to the ISO 27001 standard shows that a mapping is possible in principle, but the standards differ from slight to strong in individual points. Furthermore, it should be noted that the basis for the individual assignments of the controls was carried out and provided by the respective organization itself. Therefore, it should be noted that the individual organizations want to present their standard or norm as particularly beneficial. For example, the controls listed as partial in the gap analysis usually differ in their lack of explicit formulations that can be implicitly derived only in ISO 27001. Whether this is considered as a gap or not, depends on the point of view. Missing controls such as in the DSP (Data Security and Privacy Lifecycle Management) domain of the CCM or entire domains such as the INQ (Dealing with investigation requests from government agencies) section in the C5 catalog are different. Both examples can be covered by ISO 27001, but are not required as direct controls. These very specific, missing controls show that ISO 27001 is a more general standard on the subject of IT security and the establishment and maintenance of an ISMS. CSA STAR and BSI C5, on the contrary, are cloud specific and in some cases have a more detailed view on cloud relevant topics that play a subordinate role only in general IT security, which can be seen in the mentioned examples.

The evaluation of the three listed standards took place in the context of the de.NBI Cloud project. In order to create trust between users and the individual sites as cloud providers, a certification of all participating sites was sought. After initial attempts with the BSI C5 catalog, the decision was made in favor of a certification in accordance with ISO 27001. The reasons for this were manifold. As mentioned and shown by the mapping, most IT security or cloud specific standards are partly based on ISO 27001 and can be compared with it accordingly as it is well-known and internationally accepted. Also, the presented user survey results in Chapter 3 show that users would trust ISO 27001 more than the BSI C5 even if they do not know anything about both standards. In addition, the acquirement of further certifications would be more simple, because the ISMS as the core component would already be available through ISO 27001. Another crucial point is the risk-based approach of ISO 27001, which makes it possible to implement the predefined controls according to one's own discretion and capabilities and to create a subsequent risk assessment to determine whether a measure is sufficient or not. The CSA STAR and BSI C5 catalog make more explicit and stricter requirements for the implementation which can be helpful, but also a limiting factor. Another advantage is the availability of secondary information that argue for the ISO 27001. Of course, it is possible to describe and implement the individual controls and measures solely based on the available catalog of measures. However, if you are dealing with a certification for the first time, further secondary literature or the use of consulting companies can be helpful. Since ISO 27001 is widely used, there is a wide variety of offers.

At the end, if an organization is sure that all requirements and measures have been documented and implemented in accordance with the selected standard, an authorized auditor should be appointed to make a recommendation for or against granting a certification. Once again, the widespread use of ISO 27001 can be an advantage as many certification organizations have it in their portfolio. It can be more difficult for standards that are aligned more to the US market, such as CSA STAR. Regarding the BSI C5, instead of a certification an attestation is usually issued by a financial auditor using the catalog as a checklist.

Lessons learned

Of course issues occurred, questions were raised and valuable knowledge was acquired during the selection and implementation of a certification in the area of IT security for the de.NBI Cloud site Tübingen. The experiences made are shared within this section, including dos and don'ts.

Most of the selection process has already been described. However, in addition to the standard itself, one should also consider the feasibility of its implementation based on the available team size. All three standards presented are primarily designed for larger organizations or companies. This implies hierarchical structures, a larger number of employees and resources to implement and maintain the measures and processes of an ISMS. In the context of the de.NBI Cloud site Tübingen, the number of people, which are part of the scope was three. In the academic sector, financial and human resources and correspondingly the number of employees is usually very limited. Thus, before starting to implement a standard, it is important to check whether the specified distribution of roles can generally be implemented with a relatively thin staffing level. In the case of the cloud site Tübingen, a role system with two employees and one supervisor could be implemented, which can be regarded as a minimum. It would also be possible to have only one employee, but then the four-eye control principle would also involve the supervisor which could not possibly be suitable for everyday work. Furthermore, the top management must also be involved. This has the responsibility to promote and support the ISMS with the appropriate resources. Without the commitment of the top management, the implementation of a certification process is not advisable. As soon as the feasibility has been clarified, the documentation of processes and the implementation of measures can begin. In the scenario presented here, no prior knowledge of certification processes was available. In retrospect, a good way to get started would have been to conduct a self-assessment, for example provided by CSA's CAIQ, and to participate in beginners' workshops on the implementation of an ISMS within the framework of ISO 27001. A main focus at the beginning should also be on understanding the structure of the selected standard and its relationship to other associated standards, like ISO 27002 in case of ISO 27001. Secondary literature can also be helpful. To start with the documentation, a suitable documentation system should be used. In the case of the de.NBI cloud site Tübingen, a Confluence system[16] was operated which contains the documentation of the ISMS. In addition to features like multi-tenancy, fine-granular permission management, versioning and backups, it was beneficial that this system was already up and running, and time and effort for installation and operation could be saved. This should always be taken into account for all used auxiliary systems and software. Everything that is already there should be used if possible in order to be able to focus on the implementation of the ISMS. Further, an organizational GitLab[17] instance was used for the operational documentation, which was also already operated. A list of the software solutions used can be found in Table 6.5. All these preliminary considerations can be categorized into the ISMS planning phase. During this phase, everything that can simplify or accelerate the ISMS setup should be collected and evaluated for suitability. This is enormously important, especially in the context of a small number of employees.

In the implementation phase, the measures and resources that are already in place should be documented first. Employees should not get distracted by unfinished controls and strictly work through the individual points. The goal should not be to achieve perfection. Also stick to the language, that feels most

[16] `https://www.atlassian.com/de/software/confluence`

[17] `https://gitlab.com/gitlab-org`

Software	Application	Model
Confluence	Documentation	Proprietary
GitLab	Operational documentation	Open Source
Zabbix	Hardware monitoring	Open Source
IMOT	Asset management	Own development
Wazuh	Security Information and Event Management	Open Source
Eramba	Risk management	Proprietary

Table 6.5: List of software used to support the ISMS implementation, including the application area and what kind of licensing model is used.

comfortable, in order to avoid formulation problems or ambiguities as far as possible. This also applies to the standard itself. Understanding and interpreting a standard is already a challenge in the native language. Accordingly, the standard itself should be available in the native language of involved employees to simplify the reading process. This was also an exclusion criterion for CSA STAR since this standard was only available in English. However, attention should be paid to the choice of language before the beginning in order to provide all involved employees with the information they need in a language they can read and understand. During the documentation process, care should be taken to ensure that a clear structure exists. For the present scenario, the chapters of ISO 27001 and the annex (ISO 27002) have been adopted and used as the basic structure. Additional and required guidelines are managed in separate documents that were linked to the relevant chapters. A duplicate document structure should also be avoided in order to simplify upcoming revision processes. Also, other standards should be checked for helpful information. One example is the BSI IT-Grundschutz catalog[18], which lists 47 elementary hazards that can be used as a starting point for the risk management. Already during the setup of the ISMS, it is recommended to live and work with the system in order to become familiar with the self-imposed processes.

During the subsequent adjustment phase, the documentation and measures applied should be reviewed again. In this phase, the ISMS should already be lived in order to determine whether measures have been selected too weakly, too strongly or as not practicable at all. It does not make sense to create rules only to have people try to circumvent them or not achieve any improvement. Another important point that comes into play for the final audit is the proof of measures. An auditor will take samples from various controls and wants to see how they have been implemented and how their implementation can be verified. Accordingly, it is highly advisable to consider how each measure can be verified. In addition, a tabular listing of the measures can be used where a link to a protocol, a shell

[18] https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/IT-Grundschutz-Kompendium/Elementare-Gefaehrdungen/elementare-gefaehrdungen_node.html

command or anything else is provided to ease the verification process. An internal audit should be conducted at this stage, whether by a third party or by employees from another department for two reasons. First, this is required by ISO 27001. The corresponding audit report as well as the associated management review need to be presented to the auditor during the document review of the final certification audit. Second, new input can be gained revealing weaknesses or discrepancies leading to further adjustments and changes. These are experiences gained during the certification process specific to the de.NBI Cloud site Tübingen. However, the notes presented in this work should also be of importance for other organizations as they highlight general problems. This applies particularly for the established software, as the search for suitable software solutions can be very time-consuming.

6.6 Conclusions and Future Work

This chapter presents a concept for creating a secured virtual environment to process sensitive data. Furthermore, it includes a comparison of the IT security standards ISO 27001, CSA STAR and BSI C5 with regard to their compatibility. In addition, experiences made during a certification process according to ISO 27001 in the academic sector focused on a small number of employees were described. The content of both sections is aimed at generating trust towards cloud users. By offering a secured environment and a successful certification, a user should be given the impression that a provider, in this case the de.NBI Cloud site in Tübingen, takes IT security seriously and is suitable for processing sensitive data. It has been shown that the concept developed can reduce a wide variety of cloud specific threats. Also associated with this are minimizations of data-specific issues. However, the security of the environment can still be increased by further measures. To counter attacks and problems based on the threat of shared technology, specific hypervisors for projects working with sensitive data could be reserved for exclusive use. This would make it more difficult to carry out attacks targeting the shared RAM of a hypervisor for example. Such separation could also be performed at storage level if the available storage system offers this kind of functionality. Through the exclusive use of dedicated storage areas or storage mediums a further data separation can be achieved. Furthermore, encryption at rest can also be used on storage side to protect data even better from unauthorized access. Again, it has to be evaluated whether the storage system provides this feature or not. At network level, the use of vulnerability scanner tools like OpenVAS[19] or monitoring tools like Zabbix and Wazuh can help to detect abnormalities in the configuration of the environment and the network traffic. This would make it also possible to allow compute VMs to connect to external networks as the traffic can be monitored and

[19] https://www.openvas.org

automated measures can be configured protecting the environment. This would increase the user-friendliness for example in the context of installing software packages. Still, the compute VMs would not be accessible by the public. As a further security-enhancing measure, access to the jumphost VM can be restricted to dedicated IP addresses or address ranges. In addition to password-secured SSH keys, the usage of a two-factor authentication applying the technology of one-time passwords can further increase the security level. However, this causes an increased effort for the responsible administrator of the environment and the users have to become familiar with this technology and accept it. Without acceptance, users could get frustrated and try to circumvent the authentication mechanism leading to a reduced security of the environment.

In order to increase IT security in general, the implementation of an ISMS according to ISO 27001 was carried out. In addition to general IT security, attention should be paid to other cloud-specific certifications such as ISO 27017, CSA STAR, BSI C5 or sensitive data specific ones like ISO 27018. Especially the BSI C5 is becoming more and more acknowledged on national level, particularly in the context of hospitals. Other standards also give the opportunity to constantly expand and improve an existing ISMS because there is no perfect system.

Chapter 7

Summary and General Conclusions

During this thesis, valuable tools tackling issues in the modern paradigm of cloud computing, resource usage, virtual clusters and security have been developed. The analysis of data on tera-, petabyte scale in life sciences requires efficient bioinformatics methods and advanced compute resources. Access to powerful computing resources has become easier with the proliferation of cloud computing. However, these are valuable resources and they should be used wisely. To get an impression of how cloud resources are being used in the life sciences, a user survey has been conducted on cloud-specific topics (Chapter 3). From the results of this survey, needs and demands of users have been derived. The survey has shown an increased demand for applications measuring resource consumption on tool level. The interest relates equally to users and developers. To meet these demands, the benchmarking tool suite BOOTABLE has been developed within the scope of this thesis and evaluated by conducting a scaling study covering a wide area of bioinformatics tools, different sized datasets and compute environments (Chapter 4). Furthermore, the survey revealed that virtual clusters are of interest or already in use in addition to individual VMs. Before the widespread adoption of cloud computing, powerful computing resources were mostly available via HPC clusters. Likewise, a large number of applications tailored to the use of classical HPC clusters exists. To simplify the creation of virtual cluster environments and to take advantage of the flexibility of cloud environments, the tool VALET has been developed in this work. Furthermore, the developed and implemented scheduling algorithm has been evaluated by means of a real world example regarding its effectiveness (Chapter 5). With the increasing collection of medical personal data mostly through advanced sequencing methods, the demand for the analysis of sensitive data is rising and therefore the need for IT security as well. To enable a secure processing and analysis of such data, a security concept for the creation of virtual analysis environments using a cloud infrastructure has been developed, applied and evaluated as part of the de.NBI

Cloud project at the cloud site Tübingen. In order to gain trust of users with regard to the processing of sensitive data a process for the certification of all participating de.NBI Cloud sites has been initiated. An evaluation of possible standards and the experience made for the academic sector and small number of employees have been described in this thesis (Chapter 6). In this chapter, the individual contributions are summarized and put into relation. Finally, the thesis ends with a general conclusion.

BOOTABLE

Chapter 4 describes the bioinformatics benchmarking tool suite BOOTABLE. Since cloud computing is a relatively new technology in the life science sector, many users have little experiences with it. From the user's perspective, it looks like the available resources are endlessly available, but unfortunately they are not. With BOOTABLE it is possible to examine applications and tools for their resource consumption and scalability. During the implementation, the focus has been put on the user convenience. In addition to pre-built installation scripts, ready-to-use Docker images and selected datasets, users can choose from eleven already integrated bioinformatics applications that cover a wide range of application areas. Furthermore it is possible to integrate own tools not covered yet. Users and developers in particular were identified as the target group, but also providers to adapt hardware to the needs of customers. To cover the range of potential users, BOOTABLE collects a rich set of resource consumption metrics, but also offers the possibility to create basic reports for a quick overview.

The scaling study conducted on the eleven integrated tools revealed that the usage of more resources does not always result in better performance. Many of the applications examined seem to have a sweet spot in terms of the number of CPU cores. Furthermore, with one exception (Clustal Omega), no influence of different sized data sets on the scaling behavior of the applications could be determined. Thus, it would be possible to transfer the scaling behavior of small data sets to larger ones and perform a quick evaluation of the desired tool. In addition to different datasets, different virtual execution environments (VM, Docker) have been used to determine the resulting overhead of the virtualization layers compared to a bare-metal environment. The results have shown that the used virtualization technologies cause an overhead, which should not be underestimated. It has also been found that applications, programmed close to hardware level, had a particularly large performance loss, which can be explained by the necessary translation steps through the virtualization layer.

In summary, BOOTABLE is a benchmarking tool suite aimed at cloud users, developers and providers due to its contained features. BOOTABLE enables the efficient use of cloud resources by finding sweet spots and bottlenecks, as well as suitable hardware.

VALET

Chapter 5 presents a virtual cluster deployment tool for cloud environments, including an automated resource scheduling mechanism. Before the widespread availability of cloud computing, high performance computing resources were mostly provided as compute clusters. This resulted in applications that were optimized for cluster architectures. A (virtual) cluster is not available in a cloud environment by default unless a corresponding solution is offered by the cloud provider. This means that users are on their own. One option for them would be to cloudify their application and optimize it to cloud environments. The effort for customization can be low, high, or in some cases not feasible at all. An alternative would be to create a virtual cluster based on cloud resources. Since a successful setup of a virtual cluster requires special knowledge, that cannot be assumed by default, VALET has been developed throughout this thesis. VALET makes it possible to set up a virtual cluster automatically in less than ten minutes without requiring any special prior knowledge. As a base technology, Terraform is used to create the corresponding cloud resources (VMs, volumes, SSH keys). BeeGFS is used as a shared file system and PBS TORQUE as a batch system. In addition, the setup contains a configured instance of the middleware UNICORE including its workflow engine and the monitoring tool Zabbix. To take advantage of the flexibility of a cloud environment, a scheduling algorithm (meta scheduler) has been implemented, that enables an automated regulation of compute nodes (VMs) based on the total cluster load. To evaluate the implemented scheduling algorithm, a real world bioinformatics pipeline has been used as an example case. The results obtained have shown a resource saving potential of more than 20% for all selected check intervals compared to a static cluster. Depending on the selected parameters, users can choose an aggressive up- and down-scaling strategy or conservative variants, where for example unused resources are not immediately deleted and remain available for future use for a longer time period. By using the implemented meta scheduler, resources and also costs can be saved, which might be of interest in the case of Pay-as-You-Go models, like major cloud provider offer it. General advantages of using a virtual cluster include the fact that there is no waiting time before a job starts, since a virtual cluster is usually project specific. Furthermore, special resources such as high memory nodes or GPU nodes can be added at short notice depending on demand and availability.

With VALET, a tool is available that enables users to set up a virtual cluster conveniently and use their applications like on an HPC cluster. Furthermore, it has been shown that the provided meta scheduler is able to save resources and costs without limiting the job runtimes significantly. VALET is freely available on GitHub under `https://github.com/MaximilianHanussek/VALET`.

Security

Chapter 6 describes two topics relevant for IT security. First, a concept for a secured environment using cloud resources to process sensitive data. Second, a comparison of three IT security standards (CSA STAR, BSI C5, ISO 27001) including practical experience made during a certification process at the de.NBI Cloud site Tübingen. Security is and will become more and more important in the area of cloud computing. Attackers are becoming more professional, new security vulnerabilities are being discovered that can be exploited to disrupt cloud services, destroy or steal data. Especially with regard to sensitive data, this can have significant consequences. In order to carry out the processing of sensitive research data in a protected environment, a corresponding concept has been presented in this thesis. The security measures implemented focus on restricting public accessibility, network separation and storage access. In addition to the technical measures, the creation of a document on standard operation procedures has been introduced on organizational level, which is created and maintained by the project owner and the corresponding cloud provider. With respect to general cloud security threats (treacherous twelve, egregious eleven) as well as data specific threats, a risk reducing impact by the applied measures has been determined. In addition, the results have shown that both, cloud providers and cloud users, hold an important role in terms of accountability of security related issues. Depending on the topic, one or the other party has a greater responsibility, but mostly both remain involved.

One aim of this security concept is to protect sensitive data. Another is to generate trust with users, that the corresponding cloud provider is aware of security-related IT issues. However, concepts alone are not enough to be able to prove the security of an infrastructure and to consolidate trust. In this context, it is of interest for a cloud provider to be able to demonstrate a certification in accordance with a specific standard from the area of IT security. For the most part, the available standards are designed for larger organizations or commercial enterprises. Nevertheless, certifications are also relevant for research institutions in the academic sector. The processing of sensitive data is a major issue here as the same requirements and legislation apply just like for any other organization that processes data of this category. As part of the de.NBI Cloud project, an evaluation of potential IT security standards has been carried out. The results of the evaluation led to the decision to seek for an ISO 27001 certification. Reasons were the worldwide acceptance, the availability of numerous input sources as well as the feasibility of the implementation. Since ISO 27001 serves as a basis for many standards and is fully or partially compatible with them, further certifications can be acquired with less additional effort. An important experience made during the certification process is that an implementation with a small number of employees is difficult but possible. One outcome was that the use of appropriate software in meaningful places reduces the workload for the few people involved. Confluence and GitLab turned out to be helpful as documentation tools as well as Eramba for the risk management. In the area of monitoring, the applications Zabbix and Wazuh were used successfully. Furthermore, it should be checked whether helpful software solutions are already in use to save time and effort. Another finding was that living the system during its implementation is important to make adjustments to policies and imposed rules that were chosen too weak or too restrictive. Given enough time and a clear timeline, an IT security certification in an academic environment is possible, albeit with different problems than usual.

The presented security concept as well as the achieved ISO 27001 certification, awarded on the 4th March 2022, contribute to an increased security level regarding the processing of sensitive data as well as the general IT security of the de.NBI Cloud site Tübingen. In addition, the de.NBI Cloud site Tübingen presents itself as a trustworthy cloud provider.

General Conclusions and Outlook

To answer biological questions with bioinformatics methods, high performance computing resources are necessary. In the context of easy access to powerful compute resources, cloud computing plays an important role in the life sciences. Since resources are not infinite, they should be used efficiently and responsibly. To address this issue, the tools BOOTABLE and VALET have been implemented in this thesis. The results of a scaling study performed with BOOTABLE have shown that not all selected tools benefit from more resources. For the future, it would be of interest to extend the study with insights from applications using GPU resources. Furthermore, it has been shown that smaller data sets can be used to evaluate the general resource consumption of tools. These small data set evaluations can be used accordingly to plan resources for clusters and separated hypervisor nodes in the context of sensitive data.

In order to use cloud resources for a virtual cluster, it has to be set up first. VALET has been developed to support users with the deployment of a virtual cluster in a fast and convenient way. The results of the evaluation of the included meta scheduler have shown that a saving of resources and costs of 20% and more is possible without a high increase of the runtime values. VALET has already been used successfully in the context of a project at the de.NBI Cloud site in Tübingen [217]. In order to increase the future distribution of VALET, an integration of further batch and file systems as well as a security monitoring would be interesting, as derived from user feedback. Also a more systematic evaluation of the meta scheduler parameters, for example using AI methods, would be beneficial to cover a wider range of application scenarios. Besides the non-existing queue waiting time, a project dedicated virtual cluster offers increased security advantages. A private virtual cluster does not have to be shared with other users who do not belong to the project and thus offers more possibilities in terms of access controls. Hence, a virtual cluster would be well suited to process sensitive data. In fact, the security concept presented in Chapter 6 is similar to a virtual cluster setup. This thesis has shown that the security concept is able to minimize risks with regard to various threats specific for cloud and data. To increase the general level of IT security at the de.NBI Cloud site Tübingen, a certification was sought. The process for selecting and obtaining an ISO 27001 certification in an academic environment with a small number of employees has been outlined. In addition to ISO 27001, further cloud specific or data protection specific certifications, like ISO 27017 and ISO 27018, but also CSA STAR and BSI C5, should be considered in order to further improve the security and the public perception. However, achieving absolute security is not possible.

To conclude, this work presents tools that enable users to use cloud resources efficiently, either on application level or in context of virtual clusters. The results show that it is not always the more the merrier. In addition, the insights presented regarding cloud security concepts and the field of IT security certifications will hopefully help other organizations in academia to seek appropriate certifications and raise awareness for this important topic.

Bibliography

[1] D. A. Bader, Y. Li, and T. Li. BioPerf: A benchmark suite to evaluate high-performance computer architecture on bioinformatics applications. In *Proceedings of the 2005 IEEE International Symposium on Workload Characterization, (IISWC)*, pages 163–173, nov 2005.

[2] Amazon. Amazon Elastic Compute Cloud (Amazon EC2). `https://aws.amazon.com`. Accessed: 2021-11-30.

[3] Google. Google Cloud Computing, Hosting Services & APIs. `https://cloud.google.com/gcp/`. Accessed: 2021-11-30.

[4] Microsoft. Microsoft Azure Cloud Computing Platform; Services. `https://azure.microsoft.com`. Accessed: 2021-11-30.

[5] A. Tauch and A. Al-Dilaimi. Bioinformatics in Germany: toward a national-level infrastructure. *Briefings in Bioinformatics*, 20(2):370–374, mar 2019.

[6] J. Schulz. Überlegungen zur Steuerung einer föderativen Infrastruktur am Beispiel von bwCloud. In *Kooperation von Rechenzentren*, pages 221–242. De Gruyter, jan 2016.

[7] F. Megino, R. Jones, and K. Kucharczyk. Helix Nebula and CERN: A Symbiotic approach to exploiting commercial clouds. *Journal of Physics: Conference Series*, 513(3):32–67, jun 2014.

[8] A. Rashid and A. Chaturvedi. Cloud Computing Characteristics and Services A Brief Review. *International Journal of Computer Sciences and Engineering*, 7(2):421–426, feb 2019.

[9] R. Bhoyar and N. Chopde. Cloud Computing:Service models, types, database and issues. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(3):695–701, mar 2013.

[10] S. Rajesh, S. Swapna, and S. Reddy. Data as a Service (Daas) in Cloud Computing. *Global Journal of Computer*, 12(11):1–6, sep 2012.

[11] A. Namer. A Comparison between Cluster, Grid, and Cloud Computing. *International Journal of Computer Applications*, 179(32):37–42, apr 2018.

[12] R. Kumar and S. Charu. Comparison between Cloud Computing, Grid Computing, Cluster Computing and Virtualization. *International Journal of Modern Computer Science and Applications*, 8(31):42–47, jan 2015.

[13] C. Anderson. Docker. *IEEE Software*, 32(3):102–c3, apr 2015.

[14] G. M. Kurtzer, V. Sochat, and M. W. Bauer. Singularity: Scientific containers for mobility of compute. *PLoS ONE*, 12(5):e0177459, may 2017.

[15] T. Swathi, K. Srikanth, and S. R. Reddy. Virtualization in cloud computing. *International Journal of Computer Science and Mobile Computing*, 3(5):540–546, may 2014.

[16] M. Hanussek, F. Bartusch, and J. Krüger. Performance and scaling behavior of bioinformatic applications in virtualization environments to create awareness for the efficient use of compute resources. *PLOS Computational Biology*, 17(7):1–31, jul 2021.

[17] D. H. Ahn, J. Garlick, M. Grondona, and D. Lipari. Flux: A Next-Generation Resource Management Framework for Large HPC Centers. In *Proceedings of the International Conference on Parallel Processing Workshops*, pages 9–17, sep 2014.

[18] M. Hovestadt, O. Kao, and A. Keller. Scheduling in HPC resource management systems: Queuing vs. planning. In *Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862, pages 1–20, jun 2003.

[19] W. Zhang, A. J. Berre, D. Roman, and H. A. Huru. Migrating legacy applications to the service Cloud. In *14th Conference companion on Object Oriented Programming Systems Languages and Applications (OOPSLA)*, pages 59–68, oct 2009.

[20] K. R. Jackson, L. Ramakrishnan, and K. Muriki. Performance analysis of high performance computing applications on the Amazon Web Services cloud. In *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, (CloudCom)*, pages 159–168, feb 2010.

[21] S. Hazelhurst. Scientific computing using virtual high-performance computing: A case study using the amazon elastic computing cloud. In *ACM International Conference Proceeding Series*, volume 338, pages 94–103, oct 2008.

[22] D. Niyato, S. Chaisiri, and L. B. Sung. Optimal power management for server farm to support green computing. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, (CCGRID)*, pages 84–91, may 2009.

[23] D. Molnar and S. Schechter. Self Hosting vs . Cloud Hosting: Accounting for the security impact of hosting in the cloud. *Microsoft Research*, pages 1–18, june 2010.

[24] H. Gupta and D. Kumar. Security threats in cloud computing. In *2019 International Conference on Intelligent Computing and Control Systems, (ICCS)*, pages 1158–1162, may 2019.

[25] N. Vurukonda and B. T. Rao. A Study on Data Storage Security Issues in Cloud Computing. In *Procedia Computer Science*, volume 92, pages 128–135, jan 2016.

[26] P. G. Moreno, Y. Joly, and B. M. Knoppers. Public-private partnerships in cloud-computing services in the context of genomic research. *Frontiers in Medicine*, 4(3):1–15, jan 2017.

[27] M. Themistocleous, K. Koumaditis, and G. Vassilacopoulos. Providing integrated e-health services for personalized medicine utilizing cloud infrastructure. In *Proceedings of the European, Mediterranean and Middle Eastern Conference on Information Systems, (EMCIS)*, pages 17–18, oct 2013.

[28] M. Christodorescu, R. Sailer, and D. L. Schales. Cloud security is not (just) virtualization security: a short paper. In *Proceedings of the ACM Conference on Computer and Communications Security*, page 97–102, jan 2009.

[29] N. Tate, S. Lichtenstein, and M. J. Warren. IT security certifications: Stakeholder evaluation and selection. In *ACIS 2008 Proceedings - 19th Australasian Conference on Information Systems*, pages 991–1001, nov 2008.

[30] V. Fedák, P. Chlebana, and I. Sivỳ. IT industrial certifications in practice. In *2011 9th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 51–56. IEEE, oct 2011.

[31] I. H. Grochau, C. Schwengber ten Caten, and M. M. de Camargo Forte. Motivations, benefits and challenges on ISO/IEC 17025 accreditation of higher education institution laboratories. *Accreditation and Quality Assurance*, 23(3):183–188, jun 2018.

[32] A. Regalado. Who Coined 'Cloud Computing'? https://www.technologyreview.com/2011/10/31/257406/who-coined-cloud-computing/. Accessed: 2021-11-30.

[33] P. Mell and T. Grance. The NIST definition of cloud computing - SP 800-145. *NIST Special Publication*, 145:7, jan 2011.

[34] OpenStack. OpenStack Design. https://docs.openstack.org/arch-design/design.html. Accessed: 2021-11-30.

[35] OpenStack. General Cloud Design. https://docs.openstack.org/arch-design/use-cases/use-case-general-compute.html. Accessed 2021-11-30.

[36] M. Hasenstein. The logical volume manager (LVM). *Suse whitepaper*, jan 2001.

[37] P. Nayak and R. Ricci. Detailed study on linux logical volume manager. *Flux Research Group University of Utah*, aug 2013.

[38] RedHat Inc. GlusterFS. https://www.gluster.org. Accessed: 2021-11-30.

[39] B. Pawlowski, D. Noveck, and D. Robinson. The nfs version 4 protocol. In *Proceedings of the 2nd International System Administration and Networking Conference (SANE 2000)*, jan 2000.

[40] S. Watanabe. *Solaris 10 ZFS Essentials*. Pearson Education, jan 2009.

[41] S. A. Weil, S. A. Brandt, and E. L. Miller. Ceph: A Scalable, High-Performance Distributed File System. *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*, pages 307–320, nov 2006.

[42] Quobyte. Quobyte whitepaper. https://www.quobyte.com/resources/#whitepapers, Accessed: 2021-11-30.

[43] NetApp. NetApp whitepaper. https://www.netapp.com/pdf.html?item=/media/16905-wp-14015-na-what-are-cloud-volumes.pdf, Accessed: 2021-11-30.

[44] IBM. IBM spectrum scale whitepaper. https://www.ibm.com/support/pages/system/files/inline-files/Spectrum%20Scale%20Automation_v1.7_0.pdf, Accessed: 2021-11-30.

[45] R. Morabito, J. Kjällman, and M. Komu. Hypervisors vs. lightweight virtualization: A performance comparison. In *Proceedings - 2015 IEEE International Conference on Cloud Engineering, IC2E 2015*, pages 386–393, mar 2015.

[46] M. Hanussek, F. Bartusch, and J. Krüger. BOOTABLE: Bioinformatics benchmark tool suite for applications and hardware. *Future Generation Computer Systems*, pages 1016–1026, jan 2020.

[47] VMware. Understanding Full Virtualization, ParaVirtualization, and Hardware Assist. `https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf`. Accessed: 2021-12-17.

[48] P. Barham, B. Dragovic, and K. Fraser. Xen and the art of virtualization. In *Operating Systems Review (ACM)*, pages 164–177, dec 2003.

[49] A. Syed. Virtualization with KVM. In *Practical Linux Infrastructure*, pages 53–80. dec 2015.

[50] OpenStack. `https://www.openstack.org`. Accessed: 2021-11-30.

[51] Z. J. Estrada, Z. Stephens, and C. Pham. A performance evaluation of sequence alignment software in virtualized environments. In *Proceedings - 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, (CCGrid)*, pages 730–737, may 2014.

[52] M. Zabaljauregui. Hardware Assisted Virtualization Intel Virtualization Technology. *Unpublished Student Thesis*, jun 2008.

[53] Advanced Micro Devices Inc. AMD-V Nested Paging. `http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf`, jan 2008.

[54] J. Turnbull. *The Docker Book: Containerization is the new virtualization.* James Turnbull, jan 2014.

[55] T. Bui. Analysis of docker security. *arXiv preprint arXiv:1501.02967*, jan 2015.

[56] J. Gomes, E. Bagnaschi, and I. Campos. Enabling rootless Linux Containers in multi-user environments: The udocker tool. *Computer Physics Communications*, 232:84–97, nov 2018.

[57] F. Bartusch, M. Hanussek, and J. Krüger. Reproducible scientific workflows for high performance and cloud computing. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, jul 2019.

[58] A. Apostu, F. Puican, and G. Ularu. Study on advantages and disadvantages of cloud computing–the advantages of telemetry applications in the cloud. *Recent advances in applied computer science and digital services*, 2103:118–123, 2013.

[59] N. E. H. Bouzerzour, S. Ghazouani, and Y. Slimani. A survey on the service interoperability in cloud computing: Client-centric and provider-centric perspectives. *Software - Practice and Experience*, 50(7):1025–1060, jan 2020.

[60] S. Zhou, R. Liao, and J. Guan. When cloud computing meets bioinformatics: A review. In *Journal of Bioinformatics and Computational Biology*, volume 11, oct 2013.

[61] K. A. Shakil and M. Alam. Cloud computing in bioinformatics and big data analytics: Current status and future research. In *Advances in Intelligent Systems and Computing*, volume 654, pages 629–640, oct 2018.

[62] B. Calabrese and M. Cannataro. Cloud Computing in Bioinformatics: current solutions and challenges. Technical report, PeerJ Preprints, jul 2016.

[63] T. R. Connor, N. J. Loman, and S. Thompson. CLIMB (the Cloud Infrastructure for Microbial Bioinformatics): an online resource for the medical microbiology community. *Microbial genomics*, 2(9), sep 2016.

[64] P. Belmann, B. Fischer, and J. Krüger. de.NBI Cloud federation through ELIXIR AAI. *F1000Research*, 8, jun 2019.

[65] Z. Li, R. Ranjan, and L. O'Brien. On the Communication Variability Analysis of the NeCTAR Research Cloud System. *IEEE Systems Journal*, 12(2):1506–1517, jun 2018.

[66] A. P. Heath, M. Greenway, and R. Powell. Bionimbus: A cloud for managing, analyzing and sharing large genomics datasets. *Journal of the American Medical Informatics Association*, 21(6):969–975, jan 2014.

[67] K. Albayraktaroglu and A. Jaleel. BioBench: A benchmark suite of bioinformatics applications. In *ISPASS 2005 - IEEE International Symposium on Performance Analysis of Systems and Software*, pages 2–9, mar 2005.

[68] S. F. Altschul, W. Gish, and W. Miller. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, oct 1990.

[69] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217, sep 2000.

[70] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, nov 1994.

[71] W. R. Pearson. Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in Enzymology*, pages 63–98, mar 1990.

[72] M. Larabel and M. Tippett. Phoronix Test Suite. `https://www.phoronix-test-suite.com`, Accessed: 2021-12-04.

[73] Z. Bozkus and B. B. Fraguela. A portable high-productivity approach to program heterogeneous systems. In *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, (IPDPSW)*, pages 163–173, aug 2012.

[74] A. B. Bondi. Characteristics of scalability and their impact on performance. *Proceedings Second International Workshop on Software and Performance WOSP 2000*, pages 195–203, sep 2000.

[75] P. Berman, B. DasGupta, and M. Y. Kao. Tight approximability results for test set problems in bioinformatics. *Journal of Computer and System Sciences*, 71(2):145–162, aug 2005.

[76] D. A. Bader. Computational biology and high-performance computing. *Communications of the ACM*, 47(11):34–41, jan 2004.

[77] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, jan 2008.

[78] M. Zaharia, M. Chowdhury, and M. J. Franklin. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, jan 2010.

[79] A. Yang, M. Troup, and J. Ho. Scalability and Validation of Big Data Bioinformatics Software. *Computational and Structural Biotechnology Journal*, 15:379–386, jul 2017.

[80] D. R. Butenhof. *Programming with POSIX threads.* Addison-Wesley Professional, 1997.

[81] S. Shah and M. Bull. OpenMP. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, (SC)*, pages 13–es, nov 2006.

[82] W. Zhong, G. Altun, and X. Tian. Parallel protein secondary structure prediction schemes using Pthread and OpenMP over hyper-threading technology. *Journal of Supercomputing*, 41(1):1–16, feb 2007.

[83] C. A. MacK. Fifty years of Moore's law. In *IEEE Transactions on Semiconductor Manufacturing*, volume 24, pages 202–207, jan 2011.

[84] A. B. Craig. Science gateways for humanities, arts, and social science. In *ACM International Conference Proceeding Series*, pages 1–3, jul 2015.

[85] I. Foster, Y. Zhao, and I. Raicu. Cloud Computing and Grid Computing 360-Degree Compared. *Grid Computing Environments Workshop, (GCE)*, pages 1–10, dec 2008.

[86] R. Buyya. High performance cluster computing: Programming and applications vol. 2. *Prentice Hall PTR*, 162, 1999.

[87] F. Broquedis, J. Clet-Ortega, and S. Moreaud. hwloc: A generic framework for managing hardware affinities in HPC applications. In *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, (PDP)*, pages 180–186, feb 2010.

[88] Lustre. Lustre: A Scalable, High-Performance File System. https://cse.buffalo.edu/faculty/tkosar/cse710/papers/lustre-whitepaper.pdf, 2002.

[89] MooseFS. MooseFS. https://moosefs.com. Accessed: 2021-12-04.

[90] J. Stender, M. Berlin, and A. Reinefeld. XtreemFS: A file system for the cloud. In *Data Intensive Storage Services for Cloud Environments*, pages 267–285. jan 2013.

[91] F. Herold and S. Breuner. An introduction to BeeGFS. http://www.beegfs.io/docs/whitepapers/Introduction_to_BeeGFS_by_ThinkParQ.pdf, 2018.

[92] A. Elomari, L. Hassouni, and A. Maizate. The main characteristics of five distributed file systems required for big data: A comparative study. *Advances in Science, Technology and Engineering Systems*, 2(4):78–91, jun 2017.

[93] S. De and M. Panjwani. A Comparative Study on Distributed File Systems. pages 43–51. Springer, apr 2021.

[94] G. F. Pfister. An Introduction to the InfiniBand Architecture. *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, (42):617–632, jan 2001.

[95] M. S. Birrittella, M. Debbage, and R. Huggahalli. Enabling Scalable High-Performance Systems with the Intel Omni-Path Architecture. *IEEE Micro*, 36(4):38–47, aug 2016.

[96] G. Staples. TORQUE—TORQUE resource manager. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing, (SC)*, page 8, jan 2006.

[97] M. Jette and M. Grondona. SLURM: Simple Linux Utility for Resource Management. In *ClusterWorld Conference and Expo, (CWCE)*, volume 2682, pages 44–60, jan 2003.

[98] W. Gentzsch. Sun Grid Engine: Towards creating a compute power grid. In *Proceedings - 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, (CCGrid)*, pages 35–36, aug 2001.

[99] B. Bode, D. M. Halstead, and R. Kendall. The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 1–9, oct 2000.

[100] C. Liu, Z. Zhao, and F. Liu. An insight into the architecture of condor - A distributed scheduler. In *Proceedings - 1st International Symposium on Computer Network and Multimedia Technology, (CNMT)*, pages 1–4, jan 2009.

[101] K. Qureshi, S. Shah, and P. Manuel. Empirical performance evaluation of schedulers for cluster of workstations. *Cluster Computing*, 14(2):101–113, mar 2011.

[102] Adaptive Computing. Moab. `https://adaptivecomputing.com/moab-hpc-suite/`. Accessed: 2021-12-04.

[103] K. Benedyczak, B. Schuller, and S. Petrova-El. UNICORE 7 - Middleware services for distributed and federated computing. In *2016 International Conference on High Performance Computing and Simulation, (HPCS)*, pages 613–620, sep 2016.

[104] Y. K. Suh, H. Ryu, and H. Kim. EDISON: A Web-Based HPC Simulation Execution Framework for Large-Scale Scientific Computing Software. In *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, (CCGrid)*, pages 608–612, may 2016.

[105] M. McLennan and R. Kennell. HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering. *Computing in Science & Engineering*, 12(2):48–53, mar 2010.

[106] Amazon. AWS ParallelCluster. `https://aws.amazon.com/de/hpc/parallelcluster/`. Accessed: 2021-12-07.

[107] Kubernetes. Kubernetes. https://kubernetes.io. Accessed: 2021-12-07.

[108] ETH Zürich. ElasticCluster. https://elasticluster.readthedocs.io/en/latest/. Accessed: 2021-12-07.

[109] Microsoft. Azure Batch. https://docs.microsoft.com/azure/batch/batch-technical-overview. Accessed: 2021-12-07.

[110] P. Ruiu, O. Terzo, and G. Carlino. HPC CloudPills: On-demand deployment and execution of HPC application in cloud environments. In *Proceedings - 2014 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, (3PGCIC)*, pages 82–88, nov 2014.

[111] J. Krüger and T. Dilger. BiBiGrid. https://github.com/BiBiServ/bibigrid. Accessed: 2021-12-07.

[112] G. Juve and E. Deelman. Wrangler: Virtual cluster provisioning for the cloud. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing*, pages 277–278, jun 2011.

[113] E. Deelman, G. Singh, and M. H. Su. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, dec 2005.

[114] H. Nishimura, N. Maruyama, and S. Matsuoka. Virtual Clusters on the Fly - Fast, Scalable, and Flexible Installation. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*, pages 549–556. IEEE, may 2007.

[115] K. Keahey and T. Freeman. Contextualization: Providing one-click virtual clusters. In *Proceedings - 4th IEEE International Conference on eScience, (eScience)*, pages 301–308, dec 2008.

[116] J. Adams, M. M. Aggarwal, and Z. Ahammed. Experimental and theoretical challenges in the search for the quark-gluon plasma: The STAR Collaboration's critical assessment of the evidence from RHIC collisions. *Nuclear Physics A*, 757(1-2):102–183, aug 2005.

[117] J. S. Chase, D. E. Irwin, and L. E. Grit. Dynamic virtual clusters in a grid site manager. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing*, pages 90–100, jul 2003.

[118] I. Foster. Research Infrastructure for the Safe Analysis of Sensitive Data. *Annals of the American Academy of Political and Social Science*, 675(1):102–120, dec 2018.

[119] L. Soares, D. Fernandes, and M. Freire. Secure user authentication in cloud computing management interfaces. In *2013 IEEE 32nd International Performance Computing and Communications Conference, (IPCCC)*, pages 1–2, feb 2013.

[120] D. Abraham. Why 2FA in the cloud? *Network Security*, 2009(9):4–5, sep 2009.

[121] H. Huang, S. Baset, and C. Tang. Patch management automation for enterprise cloud. In *Proceedings of the 2012 IEEE Network Operations and Management Symposium, (NOMS)*, pages 691–705, jun 2012.

[122] H. Tabrizchi and M. Kuchaki Rafsanjani. A survey on security challenges in cloud computing: issues, threats, and solutions. *Journal of Supercomputing*, 76(12):9493–9532, feb 2020.

[123] Wazuh. `https://wazuh.com`. Accessed: 2021-12-13.

[124] IBM. QRadar. `https://www.ibm.com/downloads/cas/OP62GKAR`. Accessed: 2021-12-13.

[125] Splunk. `https://www.splunk.com/`. Accessed: 2021-12-13.

[126] Darktrace. `https://www.darktrace.com/`. Accessed: 2021-12-13.

[127] L. Kohnfelder and P. Garg. The threats to our products. *Microsoft Security Development Blog*, apr 1999.

[128] Wazuh. `https://owasp.org`. Accessed: 2022-01-06.

[129] L Chin, W. C. Hahn, and G. Getz. Making sense of cancer genomic data. *Genes and Development*, 25(6):534–555, dec 2011.

[130] D. L. Altshuler and R. M. Durbin. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061, oct 2010.

[131] L. Kacha and A. Zitouni. An Overview on Data Security in Cloud Computing. In *Advances in Intelligent Systems and Computing*, volume 661, pages 250–261, sep 2018.

[132] Z. Mahmood. Data location and security issues in cloud computing. In *Proceedings - 2011 International Conference on Emerging Intelligent Data and Web Technologies, (EIDWT)*, pages 49–54, nov 2011.

[133] D. Chen and H. Zhao. Data security and privacy protection issues in cloud computing. In *Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, (ICCSEE)*, volume 1, pages 647–651, apr 2012.

[134] R. Shaikh and M. Sasikumar. Data classification for achieving security in cloud computing. In *Procedia Computer Science*, volume 45, pages 493–498, jan 2015.

[135] H. L. Huang, Y. W. Zhao, and T. Li. Homomorphic encryption experiments on IBM's cloud quantum computing platform. *Frontiers of Physics*, 12(1):1–6, dec 2017.

[136] E. M. Mohamed, H. S. Abdelkader, and S. El-Etriby. Enhanced data security model for cloud computing. In *2012 8th International Conference on Informatics and Systems, (INFOS)*, pages CC–12, jul 2012.

[137] A. Albugmi, M. O. Alassafi, and R. Walters. Data security in cloud computing. In *2016 Fifth international conference on future generation communication technologies (FGCT)*, pages 55–59. IEEE, oct 2016.

[138] K. Iyer, R. Manisha, and R. Subhashree. Analysis of data security in Cloud Computing. In *Proceeding of IEEE - 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics, (AEEICB)*, pages 540–543, feb 2016.

[139] V. Tchifilionova. Security and privacy implications of cloud computing - Lost in the cloud. In *International Workshop on Open Problems in Network Security*, pages 149–158, jan 2010.

[140] C. Yu, L. Yang, and Y. Liu. Research on data security issues of cloud computing. In *IET Conference Publications*, volume 2014, pages 1–6, may 2014.

[141] S. M. Shariati and M. H. Ahmadzadegan. Challenges and security issues in cloud computing from two perspectives: Data security and privacy protection. In *Conference Proceedings of 2015 2nd International Conference on Knowledge-Based Engineering and Innovation, KBEI 2015*, pages 1078–1082, nov 2016.

[142] J. Huang and D. M. Nicol. Trust mechanisms for cloud computing. *Journal of Cloud Computing*, 2(1):1–14, apr 2013.

[143] S. Pape, F. Paci, and J. Jürjens. Selecting a secure cloud provider-an empirical study and multi criteria approach. *Information (Switzerland)*, 11(5):261, may 2020.

[144] C. Di Giulio, R. Sprabery, and C. Kamhoua. Cloud Standards in Comparison: Are New Security Frameworks Improving Cloud Security? In *IEEE International Conference on Cloud Computing, CLOUD*, pages 50–57, jun 2017.

[145] C. Di Giulio, R. Sprabery, and C. Kamhoua. Cloud security certifications: A comparison to improve cloud service provider security. In *ACM International Conference Proceeding Series*, pages 1–12, mar 2017.

[146] ISO 27001:2013. https://www.iso.org/standard/54534.html. Accessed: 2021-12-15.

[147] BSI C5. https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Cloud-Computing/Kriterienkatalog-C5/kriterienkatalog-c5_node.html. Accessed: 2021-12-15.

[148] CSA STAR. https://cloudsecurityalliance.org/star/. Accessed: 2021-12-15.

[149] S. S. Rizvi, T. A. Bolish, and J. R. Pfeffer. Security evaluation of cloud service providers using third party auditors. In *ACM International Conference Proceeding Series*, pages 1–6, mar 2017.

[150] E. F. Bollig, C. Henzler, and H. C. Lam. Leveraging public cloud services for clia-certified personalized medicine pipelines. In *ACM International Conference Proceeding Series*, pages 1–4, jul 2019.

[151] Clinical Laboratory Improvement Amendments. https://www.cdc.gov/clia/index.html. Accessed: 2022-01-06.

[152] S. Metz-Schimmerl, W. Schima, and C. J. Herold. Certification according to ISO 9001 - A waste of time or necessity? *Der Radiologe*, 42(5):380–386, may 2002.

[153] K. Maniam and J. Ahmad. A roadmap for the implementation of iso 9001: 2000 in academia: 9 steps to certification by fspp, uitm./maniam kaliannan and jasmine ahmad. *Malaysian Journal of Quality*, 1:83–91, jan 2005.

[154] P. Sampaio, P. Saraiva, and A. Monteiro. ISO 9001 certification pay-off: Myth versus reality. *International Journal of Quality and Reliability Management*, 29(8):891–914, aug 2012.

[155] V. V. Fomin, Y. Barlette, and H. J. de Vries. ISO/IEC 27001 Information Systems Security Management Standard : Exploring the Reasons for Low Adoption. *Proceedings of the third European conference on Management of Technology (EuroMOT)*, pages 1–13, jan 2008.

[156] M. Muñoz, J. Mejia, and A. Peña. Transitioning international software engineering standards to academia: Analyzing the results of the adoption

of ISO/IEC 29110 in four Mexican universities. *Computer Standards and Interfaces*, 66:1–27, oct 2019.

[157] Red Hat. Ansible. https://www.ansible.com. Accessed: 2021-12-15.

[158] A. El Maguiri. Openstack. *Proceedings of the Institution of Civil Engineers - Waste and Resource Management*, may 2016.

[159] N. Griffiths. nmon performance: A free tool to analyze aix and linux performance. https://developer.ibm.com/articles/au-nmon_analyser/, nov 2003.

[160] smxi. inxi. https://smxi.org/docs/inxi.html. Accessed: 2021-12-15.

[161] P. Amstutz, M. R. Crusoe, and N. Tijanić. Common Workflow Language, v1.0. *Figshare*, jul 2016.

[162] R. C. Edgar. MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, mar 2004.

[163] B. R. Brooks, C. L. Brooks, and A. D. Mackerell. CHARMM: The biomolecular simulation program. *Journal of Computational Chemistry*, 30(10):1545–1614, may 2009.

[164] D. A. Case, H. M. Aktulga, and K. Belfon. *Amber 2021*. University of California Press, jan 2021.

[165] A. Paszke, S. Gross, and F. Massa. PyTorch: An imperative style, high-performance deep learning library. *arXiv:1912.01703*, 32:8026–8037, jan 2019.

[166] J. C. Venter and M. D. Adams. The sequence of the human genome. *Science*, 291(5507):1304–1351, feb 2001.

[167] D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. 18(5):821–829, mar 2008.

[168] Y. Peng and H. C. Leung. IDBA-UD: A de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, 28(11):1420–1428, jun 2012.

[169] R. Li, H. Zhu, and J. Ruan. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, oct 2010.

[170] A. Bankevich, S. Nurk, and D. Antipov. SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology*, 19(5):455–477, may 2012.

[171] P. Medvedev, S. Pham, and M. Chaisson. Paired de Bruijn graphs: A novel approach for incorporating mate pair information into genome assemblers. *Journal of Computational Biology*, 18(11):1625–1634, nov 2011.

[172] S. I. Nikolenko, A. I. Korobeynikov, and M. A. Alekseyev. BayesHammer: Bayesian clustering for error correction in single-cell sequencing. *BMC Genomics*, 14(1):1–11, jan 2013.

[173] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor, jan 2004.

[174] B. Bushnell. BBMap: a fast, accurate, splice-aware aligner. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), mar 2014.

[175] J. Marić. Long read RNA-seq mapper. *Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu*, pages 1–62, feb 2015.

[176] B. Langmead and S. L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature methods*, 9(4):357–359, mar 2012.

[177] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, jul 2009.

[178] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 390–398, nov 2002.

[179] H. Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv 1303.3997*, may 2013.

[180] H. Li. Exploring single-sample snp and indel calling with whole-genome de novo assembly. *Bioinformatics*, 28(14):1838–1844, may 2012.

[181] P. Bawono, M. Dijkstra, and W. Pirovano. Multiple sequence alignment. In *Methods in Molecular Biology*, pages 167–189. nov 2017.

[182] F. Sievers and D. G. Higgins. Clustal Omega. *Current Protocols in Bioinformatics*, 48(1):3–13, dec 2014.

[183] F. Sievers and D. G. Higgins. Clustal Omega, Accurate Alignment of Very Large Numbers of Sequences. In *Methods in Molecular Biology*, pages 105–116. dec 2014.

[184] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, jun 2006.

[185] J. Söding. Protein homology detection by HMM-HMM comparison. *Bioinformatics*, 21(7):951–960, nov 2005.

[186] F. Sievers, A. Wilm, and D. Dineen. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1):539, sep 2011.

[187] K. Katoh, K. Misawa, and K. I. Kuma. MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, jul 2002.

[188] K. Katoh, K. I. Kuma, and H. Toh. MAFFT version 5: Improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, 33(2):511–518, jan 2005.

[189] K. Katoh and H. Toh. Recent developments in the MAFFT multiple sequence alignment program. *Briefings in Bioinformatics*, 9(4):286–298, 2008.

[190] K. Katoh and H. Toh. Parallelization of the MAFFT multiple sequence alignment program. *Bioinformatics*, 26(15):1899–1900, apr 2010.

[191] K. Katoh and D. M. Standley. MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution*, 30(4):772–780, apr 2013.

[192] E. Pruesse, J. Peplies, and F. O. Glöckner. SINA: Accurate high-throughput multiple sequence alignment of ribosomal RNA genes. *Bioinformatics*, 28(14):1823–1829, jul 2012.

[193] C. Quast, E. Pruesse, and P. Yilmaz. The SILVA ribosomal RNA gene database project: Improved data processing and web-based tools. *Nucleic Acids Research*, 41(D1):D590–D596, jan 2013.

[194] W. Ludwig, O. Strunk, and Westram R. ARB: A software environment for sequence data. *Nucleic Acids Research*, 32(4):1363–1371, feb 2004.

[195] C. Lee, C. Grasso, and M. F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, mar 2002.

[196] M. J. Abraham, T. Murtola, and R. Schulz. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*, 1:19–25, jul 2015.

[197] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, aug 1993.

[198] M. Abadi, P. Barham, and J. Chen. Tensorflow: A system for large-scale machine learning. *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, nov 2016.

[199] CIFAR. CIFAR-10. `https://www.tensorflow.org/tutorials/images/cnn`. Accessed: 2021-12-16.

[200] K. He and J. Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5353–5360, jun 2015.

[201] A. Auton and T. Salcedo. The 1000 genomes project. In *Assessing Rare Variation in Complex Traits: Design and Analysis of Genetic Studies*, pages 71–85. jan 2015.

[202] R. Leinonen, H. Sugawara, and M. Shumway. The sequence read archive. *Nucleic Acids Research*, 39(suppl_1):D19–D21, jan 2011.

[203] E. W. Sayers, M. Cavanaugh, and K. Clark. GenBank. *Nucleic Acids Research*, 47(D1):D94–D99, jan 2019.

[204] H. C. Bernstein, C. J. Brislawn, and K. Dana. Primary and heterotrophic productivity relate to multikingdom diversity in a hypersaline mat. *FEMS Microbiology Ecology*, 93(10):1–10, oct 2017.

[205] B. Hess, C. Kutzner, and D. Van Der Spoel. GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of Chemical Theory and Computation*, 4(3):435–447, feb 2008.

[206] S. Páll and B. Hess. A flexible algorithm for calculating pair interactions on SIMD architectures. *Computer Physics Communications*, 184(12):2641–2650, jun 2013.

[207] D. Yu, J. Wang, and B. Hu. A practical architecture of cloudification of legacy applications. In *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, pages 17–24, sep 2011.

[208] M. A. Netto, R. N. Calheiros, and E. R. Rodrigues. HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges. *ACM Computing Surveys*, 51(1):1–29, jan 2018.

[209] E. Tantoso, W. C. Wong, and W. H. Tay. Hypocrisy Around Medical Patient Data: Issues of Access for Biomedical Research, Data Quality, Usefulness for the Purpose and Omics Data as Game Changer. *Asian Bioethics Review*, 11(2):189–207, jun 2019.

[210] K. V. Voelkerding, S. Dames, and J. D. Durtschi. Next Generation Sequencing for Clinical Diagnostics-Principles and Application to Targeted Resequencing for Hypertrophic Cardiomyopathy. *The Journal of Molecular Diagnostics*, 12(5):539–551, dec 2010.

[211] S. D. Kahn. On the future of genomic data. *Science*, 331(6018):728–729, feb 2011.

[212] M. Pfeiffer, M. Rossberg, and S. Buttgereit. In *Proceedings of the 14th International Conference on Availability, Reliability and Security.*

[213] ThinkParQ. BeeOND. https://www.beegfs.io/wiki/BeeOND. Accessed: 2021-12-16.

[214] OpenStack. Manila. https://wiki.openstack.org/wiki/Manila. Accessed: 2021-12-16.

[215] E. Afgan, D. Baker, and B. Batut. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, 46(W1):W537–W544, may 2018.

[216] P. Di Tommaso, M. Chatzou, and E. W. Floden. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, apr 2017.

[217] D. Porubsky, P. Ebert, and P. A. Audano. Fully phased human genome assembly without parental data using single-cell strand sequencing and long reads. *Nature Biotechnology*, 39(3):302–308, dec 2020.

[218] P. Swierczynski, G. Leander, and C. Paar. Keccak und der SHA-2. *Datenschutz und Datensicherheit - DuD*, 37(11):712–719, nov 2013.

[219] The practical obstacles of data transfer: Why researchers still love scp. In *Proceedings of the Third International Workshop on Network-Aware Data Management*, pages 1–8, nov 2013.

[220] A. Tridgell and P. Mackerras. The rsync algorithm. *Joint Computer Science Tecnical Report Series*, TR-CS-96-0(June):1–6, june 1996.

[221] L. Dai, X. Gao, and Y. Guo. Bioinformatics clouds for big data manipulation. *Biology Direct*, 7(1):1–7, nov 2012.

[222] A. Choudhary and R. Bhadada. Emerging Threats in Cloud Computing. In *Communications in Computer and Information Science*, volume 1214, pages 147–156, jul 2020.

[223] M. D. Hill, J. Masters, and P. Ranganathan. On the Spectre and Meltdown Processor Security Vulnerabilities. *IEEE Micro*, 39(2):9–19, feb 2019.

Appendix A

Supporting Information

BOOTABLE Parameter List

The following list is a complete description of the available parameters for BOOTABLE.

- Clean (-c): If this parameter is set the benchmark tool takes care of already executed and finished benchmarks. The results of existing benchmarks will be backed up and all unnecessary temporary data are deleted.

- Dataset (-d):Three dataset combinations can be selected using the keywords small, medium and large with regard to size and complexity. The larger the dataset the longer the runtime of a benchmark.

- Generic (-o): This option provides the possibility to include tools not already integrated into the benchmark suite, using a simple configuration file. The file path to the configuration file is required here.

- Threads (-p): As all of the used benchmark applications are multithreaded any reasonable number of threads can be chosen that the applications should use in parallel. This option is applied to all tools started in a benchmark run. As before the keywords full, half and one are provided. Full will take all available cores, half the half of the available cores and one a single CPU core.

- Replicates (-r): Bioinformatics applications sometimes work with random number seeds as a starting point as some problems can only be solved in reasonable time through approximations due to their computational complexity. In order to adjust the impact of outliers regarding the runtime it is recommended to run the benchmarks at least three times, which are called replicates.

- Scaling (-s): In order to give application developers or users the possibility to examine a given tool during the development phase or get an understanding of the resource consumption limits, a so-called scaling mode is provided. The implemented scaling mode runs an automated sequence of different thread sizes.

- Toolgroup (-t): The here presented benchmark applications widely differ in their runtime behavior and the underlying computational calculations. Therefore only subsets of specific tools might be of interest. These subsets can be chosen with the toolgroup parameter which accepts the keywords all, genomics, quant and ml. Further is it possible to pick just one single tool by its name.

Full Domain Names CSA CCM

Abbreviation	Full Name
A&A	Audit & Assurance
AIS	Application & Interface Security
BCR	Business Continuity Management and Operational Resilience
CCC	Change Control and Configuration Management
CEK	Cryptography, Encryption & Key Management
DCS	Datacenter Security
DSP	Data Security and Privacy Lifecycle Management
GRC	Governance, Risk and Compliance
HRS	Human Resources
IAM	Identity & Access Management
IPY	Interoperability & Portability
IVS	Infrastructure & Virtualization Security
LOG	Logging and Monitoring
SEF	Security Incident Management, E-Discovery, & Cloud Forensics
STA	Supply Chain Management, Transparency, and Accountability
TVM	Threat & Vulnerability Management
UEM	Universal Endpoint Management

Table A.1: Mapping of CSA CCM domain abbreviations to full names.

Full Domain Names BSI C5

Abbreviation	Full Name
OIS	Organization of Information Security
SP	Security Policies and Instruction
HR	Personnel
AM	Asset Management
PS	Physical Security
OPS	Operations
IDM	Identity and Access Management
CRY	Cryptography and Key Management
COS	Communication Security
PI	Portability and Interoperability
DEV	Procurement, Development and Modification of Information Systems
SSO	Control and Monitoring of Service Providers and Suppliers
SIM	Security Incident Management
BCM	Business Continuity Management
COM	Compliance
INQ	Dealing with investigation requests from government agencies
PSS	Product Safety and Security

Table A.2: Mapping of BSI C5 domain abbreviations to full names.

Appendix B

Software Licenses and Availability

Toolname	License	Availability
BBMap	Open source and free for unlimited use	https://jgi.doe.gov/data-and-tools/software-tools/bbtools/
BeeOND	BeeGFS End User License Agreement	https://doc.beegfs.io/latest/advanced_topics/beeond.html
Biobench2	Apache 2.0 License	https://github.com/reiverjohn/biobench2
Openbenchmarking	GPL-3.03	https://openbenchmarking.org
BOOTABLE Docker Image	-	https://hub.docker.com/r/maximilianhanussek/bootable
BOOTABLE Tool	-	https://github.com/MaximilianHanussek/BOOTABLE
Bowtie2	GPL-3.0	https://github.com/BenLangmead/bowtie2
BWA	GPL-3.0	https://github.com/lh3/bwa
CIFAR-10 model build	Apache-2.0 License	https://www.tensorflow.org/tutorials/images/cnn
Clustal Omega	GPL-2.0	http://www.clustal.org/omega/
GROMACS	LGPL	https://manual.gromacs.org/current/download.html
IDBA-UD	GPL	https://github.com/loneknightpy/idba
MAFFT	BSD/GPL	https://mafft.cbrc.jp/alignment/software/
nmon	GPL-3.0	https://nmon.sourceforge.net/pmwiki.php?n=Site.CompilingNmon
PBS Torque	Proprietayr License / OpenPBS v2.3	https://github.com/adaptivecomputing/torque
SINA	GPL-3.0	https://sina.readthedocs.io/en/latest/
SPAdes	GPL-2.0	https://github.com/ablab/spades
UNICORE	BSD	https://www.unicore.eu
VALET	-	https://github.com/MaximilianHanussek/VALET
Velvet	GPL-2.0	https://github.com/dzerbino/velvet
Zabbix	GPL-2.0	https://www.zabbix.com

Table B.1: Listing of used software and tools with reference to their license as well as their availability.

Appendix C

Publications

Accepted manuscripts

2021

M. Hanussek, F. Bartusch, J. Krüger, Performance and scaling behavior of bioinformatic applications in virtualization environments to create awareness for the efficient use of compute resources, PLOS Comput. Biol. 17 (2021). doi:10.1371/journal.pcbi.1009244.

2020

M. Hanussek, F. Bartusch, J. Krüger, BOOTABLE: Bioinformatics benchmark tool suite for applications and hardware, Futur. Gener. Comput. Syst. 102 (2020). doi:10.1016/j.future.2019.09.057.

R. Rajan, K.P. Divya, R.M. Kandadai, R. Yadav, V.P. Satagopam, U.K. Madhusoodanan, P. Agarwal, N. Kumar, T. Ferreira, H. Kumar, A.V.S. Prasad, K. Shetty, S. Mehta, S. Desai, S. Kumar, L.K. Prashanth, M. Bhatt, P. Wadia, S. Ramalingam, G.M. Wali, S. Pandey, F. Bartusch, **M. Hanussek**, J. Krüger, A. Kumar-Sreelatha, S. Grover, P. Lichtner, M. Sturm, J. Roeper, V. Busskamp, G.R. Chandak, J. Schwamborn, P. Seth, T. Gasser, O. Riess, V. Goyal, P.K. Pal, R. Borgohain, R. Krüger, A. Kishore, M. Sharma, Genetic architecture of parkinson's disease in the indian population: Harnessing genetic diversity to address critical gaps in parkinson's disease research, Front. Neurol. 11 (2020). doi:10.3389/fneur.2020.00524.

2019

P. Belmann, B. Fischer, J. Krüger, M. Procházka, H. Rasche, M. Prinz, **M. Hanussek**, M. Lang, F. Bartusch, B. Gläßle, J. Krüger, A. Pühler, A. Sczyrba, de. NBI cloud federation through ELIXIR AAI [version 1; peer review: 2 approved, 1 not approved], F1000Research. 8 (2019). doi:10.12688/f1000research.19013.1.

M. Hanussek, F. Bartusch, J. Kruger, O. Kohlbacher, BOOTABLE: Bioinformatics benchmark tool suite, in: Proc. - 19th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGrid 2019, 2019. doi:10.1109/CCGRID.2019.00027.

F. Bartusch, **M. Hanussek**, J. Kruger, O. Kohlbacher, Reproducible scientific workflows for high performance and cloud computing, in: Proc. - 19th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput. CCGrid 2019, 2019. doi:10.1109/CCGRID.2019.00028.

B. Gläßle, **M. Hanussek**, F. Bartusch, V. Lutz, U. Hahn. W. Dilling, T. Walter, J. Krüger, de. NBI Cloud Storage Tübingen. A federated and georedundant solution for large scientific data. (2019).

2018

F. Bartusch, **M. Hanussek**, J. Krüger, Automatic generation of provenance metadata during execution of scientific workflows, in: CEUR Workshop Proc., 2018.

2017

M. Hanussek, F. Bartusch, J. Krüger, O. Kohlbacher, Efficient mass spectra prediction through container orchestration with a scientific workflow, in: CEUR Workshop Proc., 2017.

Appendix D

Contributions co-authorship

Author	Author position	Scientific ideas %	Data generation %	Analysis & interpretation %	Paper writing %
Maximilian Hanussek	First author	95	100	95	100
Felix Bartusch	Second author	5	0	0	0
Jens Krüger	Corresponding author	0	0	5	0
Title of paper:		BOOTABLE: Bioinformatics benchmark tool suite for applications and hardware			
Status in publication process:		Published			

Table D.1: Contributions of co-authors for the publication: Bioinformatics benchmark tool suite for applications and hardware.

Author	Author position	Scientific ideas %	Data generation %	Analysis & interpretation %	Paper writing %
Maximilian Hanussek	First author	100	100	90	100
Felix Bartusch	Second author	0	0	5	0
Jens Krüger	Corresponding author	0	0	5	0
Title of paper:		Performance and scaling behavior of bioinformatic applications in virtualization environments to create awareness for the efficient use of compute resources			
Status in publication process:		Published			

Table D.2: Contributions of co-authors for the publication: Performance and scaling behavior of bioinformatic applications in virtualization environments to create awareness for the efficient use of compute resources.

Eidesstattliche Versicherung

Ich erkläre hiermit, dass ich die zur Promotion eingereichte Arbeit mit dem Titel: Cloud Computing in Bioinformatics: Benchmarking, Virtual Cluster, Security selbständig verfasst, nur die angegebenen Quellen und Hilfsmittel benutzt und wörtlich oder inhaltlich übernommene Stellen (alternativ: Zitate) als solche gekennzeichnet habe. Ich erkläre, dass die Richtlinien zur Sicherung guter wissenschaftlicher Praxis der Universität Tübingen (Beschluss des Senats vom 25.5.2000) beachtet wurden. Ich versichere an Eides statt, dass diese Angaben wahr sind und dass ich nichts verschwiegen habe. Mir ist bekannt, dass die falsche Abgabe einer Versicherung an Eides statt mit Freiheitsstrafe von bis zu drei Jahren oder mit Geldstrafe bestraft wird.

Ort, Datum   Unterschrift