Sjaak Laan

IT Infrastructure Architecture

Infrastructure Building Blocks and Concepts

4th Edition

Title: **IT Infrastructure Architecture –**
Infrastructure Building Blocks and Concepts
4th Edition

Author: **Sjaak Laan**

Publisher: **Lulu Press Inc.**

ISBN: **978-1-4477-8560-6**

Edition: **4th edition, 2023**

Trademarks

All trademarks used in this book are the property of their respective owners.

- AIX is a trademark of IBM Corp., registered in the U.S. and other countries.
- ArchiMate is a registered trademark of The Open Group.
- AWS (Amazon Web Services) is a trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.
- AMD Opteron, the AMD logo, the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices.
- Apache®, Apache Tomcat, and Apache Mesos are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.
- Apple, Mac, iOS, and Mac OS are trademarks of Apple Inc., registered in the U.S. and other countries.
- Cisco is a registered trademark of Cisco in the U.S. and other countries.
- Citrix, XenServer, XenMotion XenServer Marathon everRun, MetaFrame Presentation Server, XenApp, and XenDesktop are trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.
- DEC™, DECnet™, VMS™, and VAX™ are trademarks of Digital Equipment Corporation.
- Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.
- Gartner Hype Cycle is a registered trademark of Gartner, Inc. and/or its affiliates and is used herein with permission. All rights reserved.

- Google, Android, Google App Engine, and Kubernetes are registered trademarks of Google Inc
- HP and HPE are a registered trademark of Hewlett-Packard Company in the U.S. and other countries.
- IBM, AIX, IBM MQ, DB2, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, and/or other countries.
- Intel, Intel Core, Xeon, and Thunderbolt are trademarks of Intel Corp. in the U.S. and other countries.
- IOS is a trademark or registered trademark of Cisco in the U.S. and other countries.
- Java and all Java-based trademarks are trademarks of Oracle, Inc. in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds.
- Microsoft®, Hyper-V, Windows, Windows NT®, Microsoft Azure Cloud Service, Windows .Net, Microsoft Internet Information Services, BizTalk, Microsoft SQL Server, and the Windows logo are trademarks of Microsoft Corporation in the United States and other countries.
- Oracle, Sun Microsystems, and Java are registered trademarks of Oracle Corporation and/or its affiliates.
- The Pivotal CloudFoundry trademark is the property of Pivotal Software, Inc. and its subsidiaries and affiliates (collectively "Pivotal").
- PowerPC™ and the PowerPC logo™ are trademarks of International Business Machines Corporation.
- Red Hat Enterprise Linux and Red Hat JBoss are trademarks of Red Hat, Inc. in the United States and other countries.
- SPEC® is a registered trademark of the Standard Performance Evaluation Corporation (SPEC). See http://www.spec.org for more information.
- TOGAF is a registered trademark of The Open Group in the United States and other countries.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

- VMware, VMware tools, VMware Workstation, VMware Fault Tolerance, Sphere, GSX, ESX, ESXi, vCenter, and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

Other company, product, or service names may be trademarks or service marks of others. All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized.

While every precaution was made in the preparation of this book, the author can assume no responsibility for errors or omissions. If you feel the author has not given you proper credit or feel your rights were violated, please notify the author so corrective actions can be taken.

Pictures used in this book are created by the author of this book or are freely distributable pictures, retrieved from the internet. Most of the used pictures are from the public domain. When a picture is used that contained copyrights, a link to the source of the picture and its copyright notice is provided. If you feel a picture used in this book is not freely distributable, or any other copyright is violated, please inform the author, so it can be corrected in the next version of the book.

Table of Contents

Preface

What this book is about

This book is about information technology (IT) infrastructure architecture. Infrastructure refers to all the hardware and system software components required to run IT applications. And infrastructure architecture describes the overall design and evolution of that infrastructure.

This book explains how infrastructure components work at the architectural level. This means that components are described in building blocks that are tied to specific infrastructure technologies. Decisions made at this level are architecturally relevant, which means that once decisions are made at the building block level, it is relatively difficult to change them later. For example, the decision to use a particular cabling infrastructure in a datacenter cannot be easily changed once the datacenter is in operation.

This book does not provide the level of detail required by engineers, but rather describes the most important architectural building blocks and concepts.

IT infrastructures are complex by nature and provide non-functional attributes such as performance, availability, and security to applications. This book describes each infrastructure building block and its specific performance, availability, and security concepts.

Until now, there has been no single publication that describes the entire field of IT infrastructure. Books and papers exist on each part of IT infrastructure, such as networking, installing and managing operating systems, storage, and virtualization, but no publication has yet described IT infrastructure as a whole. This book aims to fill that gap.

Intended audience

This book is intended for infrastructure architects and designers, software architects, systems managers, and IT managers. It can also be used in education, for example in a computer science class. This book is very suitable for beginners, as almost every term is explained, while for experts and professionals this book is more of a review and overview.

Infrastructure architects and designers can use this book to learn more about infrastructure design that is not their core competency. For example, network designers will probably not learn anything new about networking, but they will probably learn a lot about all the other parts of the infrastructure, such as datacenters, storage, and servers. The same is true for other designers.

Software architects build software that runs on infrastructure. Software architects who understand the challenges an infrastructure architect faces can optimize their software for certain infrastructure characteristics. Understanding infrastructure helps software architects build more reliable, faster, manageable, and secure applications.

Systems managers learn to identify key architectural choices and principles in an infrastructure, as well as ways to update and change a running infrastructure without compromising the architecture as a whole.

IT managers gain a complete view of IT infrastructures and IT architecture. This will help them work with system administrators and infrastructure architects to better understand their concerns.

Students of computer science will find a wealth of information about IT infrastructures that will provide a solid foundation for their computer science studies. This book is used by a number of universities around the world as part of their IT architecture curricula. It is particularly suitable for courses based on the Association for Computing Machinery (ACM) IS 2020.3 curriculum. A reference matrix of the curriculum topics and the relevant sections in this book is provided in the appendix *IS 2020.3 Curriculum reference matrix*.

Some basic IT knowledge is needed to read this book, but the reader is introduced to each topic in small steps.

Acknowledgements

I would like to thank my wife, Angelina, for the patience she showed when I was working again on this book for a whole evening or weekend, without giving her the attention she deserves, and my three children Laura, Maarten, and Andreas, who I love.

Jan van Til inspired me to think more thoroughly about the definition of infrastructure. His (Dutch) work on information management can be found at www.emovere.nl.

I want to thank Robert Elsinga, Olav Meijer, Esther Barthel, Raymond Groenewoud, Emile Zweep, Cathy Ellis, Jacob Mulder, Robbert Springer, Marc Eilander, and Jan van der Zanden for their criticism, useful suggestions, and hard work when reviewing this book.

For the transformation of this book to the online training on MyEducator, I would like to thank VP Sales Scott Pectol and Content Team Lead Aspen Moore.

Especially I want to thank Lodewijk Bogaards, who reviewed the book's first edition and provided literally hundreds of useful tips on the described topics. He also made many corrections on my English grammar.

The photo on the cover is based on the Shutterstock photo https://www.shutterstock.com/image-photo/server-rack-cluster-data-center-shallow-71676715, taken by a photographer named Lightpoet.

Courseware

This book is used in a number of universities around the world as a resource for their IT infrastructure courses. For more information about using this book in a university course, please contact the author at **sjaak.laan@gmail.com**.

Courseware can be downloaded from **www.sjaaklaan.com/book**. It contains all the figures used in the book in both Visio and high-resolution PNG format, the list of abbreviations, a PowerPoint slide deck for each chapter (over 700 slides in total), a set of test questions for each chapter (over 200 questions in total), and the infrastructure checklist from the appendix.

This book is also available as online training from MyEducator. Please check **https://app.myeducator.com/reader/web/1957a/**.

Note to the fourth edition

In the fourth edition of this book, a number of corrections have been made, some terminology has been clarified, and several typographical and syntax errors have been corrected. In addition, the following changes have been made:

- The content has been updated to reflect the new Association for Computing Machinery (ACM) IS 2020.3 Curriculum - Competency Area - IT Infrastructure.
- A new chapter on cloud computing has been added, and cloud-related content has been added throughout the rest of the book.
- A new chapter on documenting infrastructures was added.
- New technologies such as serverless computing, edge computing and quantum computing have been added.
- The security chapter has been rewritten and restructured to better reflect infrastructure-related security concerns.
- The Infrastructure as Code chapter has been rewritten to reflect current working practices and a chapter on automation has been added as this has become more important over the years.
- The chapter on Purchasing Infrastructure and Services has been removed as it was too general and not specific to infrastructure. The chapter was mandatory for the IS 2010.4 syllabus, but has been removed from the IS 2020.3 syllabus.
- The networking chapter has been expanded to include POP, SMTP, FTP, HTTP, and HTTPS protocols. This is a requirement from the IS 2020.3 syllabus.
- An appendix has been added that describes a high-level checklist that can be used to ask the right questions when learning about an existing infrastructure in the field.
- More than 100 edits were made throughout the book to clarify and update content, and to remove outdated content.
- Finally, as technology has advanced in recent years, the book has been updated to include the most current information.

About the Author

Sjaak Laan (1964) leads CGI's Cloud and Infrastructure practice in the Netherlands. After studying electronics in the 1980s, he started his career in the IT industry at a PC repair company, where he repaired thousands of IBM PS/2 system boards at chip level. He later became an IT infrastructure specialist in networking, storage and computing. He now has more than 30 years of IT experience.

Mr. Laan joined CGI in 2000 and is now a Director Consulting Expert in the government, financial, and energy markets. He is an expert in cloud, infrastructure and security and has extensive knowledge of systems management processes and integrations.

As an architect, he is certified by The Open Group as a Master IT Architect and is TOGAF certified. In the area of cloud, he is an AWS Certified Solution Architect and Certified Azure Solutions Architect Expert. His information security knowledge is supported by his CISSP and CRISC certifications.

Sjaak Laan has been writing about cloud and infrastructure on www.sjaaklaan.nl since 2006, has a number of publications to his name and regularly gives trainings and presentations. Mr. Laan usually works for clients as a lead architect or consultant on complex projects.

PART I
-
INTRODUCTION TO IT INFRASTRUCTURE

Infrastructure is much more important than architecture.

Rem Koolhaas, one of the world's most famous architects

1

THE DEFINITION OF IT INFRASTRUCTURE

1.1 Introduction

In the early decades of IT development, most infrastructures were relatively simple. As applications grew in functionality and complexity, hardware basically just got faster. In recent years, IT infrastructures have become more complex due to the rapid development and deployment of new types of applications, such as big data, artificial intelligence (AI), machine learning, the Internet of Things (IoT), and cloud computing. These applications require new and more sophisticated infrastructure services that are secure, highly scalable, and available 24/7.

1.2 What is IT infrastructure?

IT infrastructure has been around for a long time. But surprisingly, there does not seem to be a universally accepted definition of IT infrastructure. I have found that many people are confused by the term IT infrastructure, and a clear definition would help them understand what IT infrastructure is and is not.

In literature, many definitions of IT infrastructure can be found. Some of them are:

- IT infrastructure is defined broadly as a set of information technology (IT) components that are the foundation of an IT service; typically

physical components (computer and networking hardware and facilities), but also various software and network components.
Wikipedia

- All of the hardware, software, networks, facilities, etc., that are required to develop, test, deliver, monitor, control, or support IT services. The term IT Infrastructure includes all of the Information Technology but not the associated people, processes and documentation.
ITILv3.

- IT infrastructure refers to the composite hardware, software, network resources and services required for the existence, operation and management of an enterprise IT environment. IT infrastructure allows an organization to deliver IT solutions and services to its employees, partners and/or customers and is usually internal to an organization and deployed within owned facilities.
Techopedia

- IT infrastructure is the system of hardware, software, facilities and service components that support the delivery of business systems and IT-enabled processes.
Gartner

- IT infrastructure refers to the combined components needed for the operation and management of enterprise IT services and IT environments.
IBM

- IT infrastructure are the components required to operate and manage enterprise IT environments. IT infrastructure can be deployed within a cloud computing system, or within an organization's own facilities. These components include hardware, software, networking components, an operating system (OS), and data storage, all of which are used to deliver IT services and solutions.
Red Hat

Based on these definitions, the term infrastructure may seem a bit arbitrary. Let's try to clear things up.

The word infrastructure comes from the words infra (Latin for "underneath") and structure. It encompasses all the components that are "underneath" the structure, where the structure may be a city, a house, or an information system. In the physical world, infrastructure often refers to public utilities such as water pipes, power lines, gas pipes, sewers, and telephone lines – components that literally lie beneath the structure of a city.

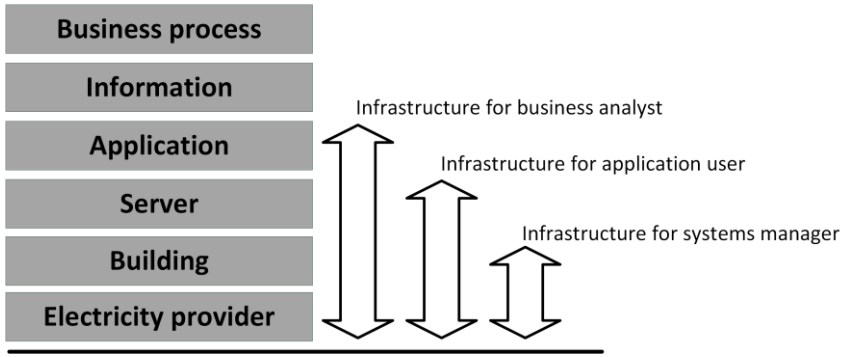


Figure 1: Views on IT infrastructure

For most people, infrastructure is invisible and taken for granted. When a business analyst describes business processes, the information used in the process is very important. How that information is managed by IT systems is "below the surface" to the business analyst. They think of IT systems as infrastructure.

For users of IT systems, applications are important because they use them every day, but how they are implemented or where they are physically located is invisible (below the surface) to them and is therefore considered infrastructure.

For systems managers, the building that houses their servers and the utility company that provides the power are considered infrastructure.

So what infrastructure is depends on who you ask and their point of view.

The scope of infrastructure as used in this book is described in more detail in chapter 2.

1.3 What is IT architecture?

Most of today's infrastructure landscapes are the result of a history of application implementation projects that brought in their own specialized hardware and infrastructure components. Mergers and acquisitions have made matters worse, leaving many organizations with multiple sets of the same infrastructure services that are difficult to interconnect, let alone integrate and consolidate.

Organizations benefit from infrastructure architecture when they want to be more flexible and agile because a solid, scalable, and modular infrastructure provides a solid foundation for agile adaptations. The market demands a level of agility that can no longer be supported by infrastructures that are inconsistent and difficult to scale. We need infrastructures built with standardized, modular

components. And to make infrastructures consistent and aligned with business needs, architecture is critical.

Architecture is the philosophy that underlies a system and defines its purpose, intent, and structure. Different areas of architecture can be defined, including business architecture, enterprise architecture, data architecture, application architecture, and infrastructure architecture. Each of these areas has certain unique characteristics, but at their most basic level, they all aim to map IT solutions to business value.

Architecture is needed to govern an infrastructure as it is designed, as it is used, and as it is changed. We can broadly categorize architects into three groups: enterprise architects, domain architects, and solution architects, each with their own role.

1.3.1 Solution architects

Solution architects create IT solutions, usually as a member of a project team. A solution architect is finished when the project is complete. Solution architects are the technical conscience and authority of a project, are responsible for architectural decisions in the project, and work closely with the project manager.

Where the project manager manages the *process* of a project, the solution architect manages the technical *solution* of the project, based on business and technical requirements.

1.3.2 Domain architects

Domain architects are experts on a particular business or technology topic. Because solution architects cannot always be fully knowledgeable about all technological details or specific business domain issues, domain architects often assist solution architects on projects. Domain architects also support enterprise architects because they are aware of the latest developments in their field and can inform enterprise architects about new technologies and roadmaps. Examples of domain architects are cloud architects, network architects, and VMware architects.

Domain architects most often work for infrastructure or software vendors, where they help customers implement the vendor's technologies.

1.3.3 Enterprise architects

Enterprise architects continuously align an organization's entire IT landscape with the business activities of the organization. Using a structured approach, enterprise architects enable transformations of the IT landscape (including the

IT infrastructure). Therefore, an enterprise architect is never finished (unlike the solution architect in a project, who is finished when the project is finished).

Enterprise architects typically work closely with the CIO and business units to align the needs of the business with the current and future IT landscape. Enterprise architects build bridges and act as advisors to the business and IT.

2

THE INFRASTRUCTURE MODEL

2.1 IT building blocks

The definition of infrastructure as used in this book is based on the building blocks in the model as shown in Figure 2. In this model, processes consume information, and that information is stored and managed by applications. Applications require application platforms and infrastructure to run. All of this is managed by different categories of systems management.

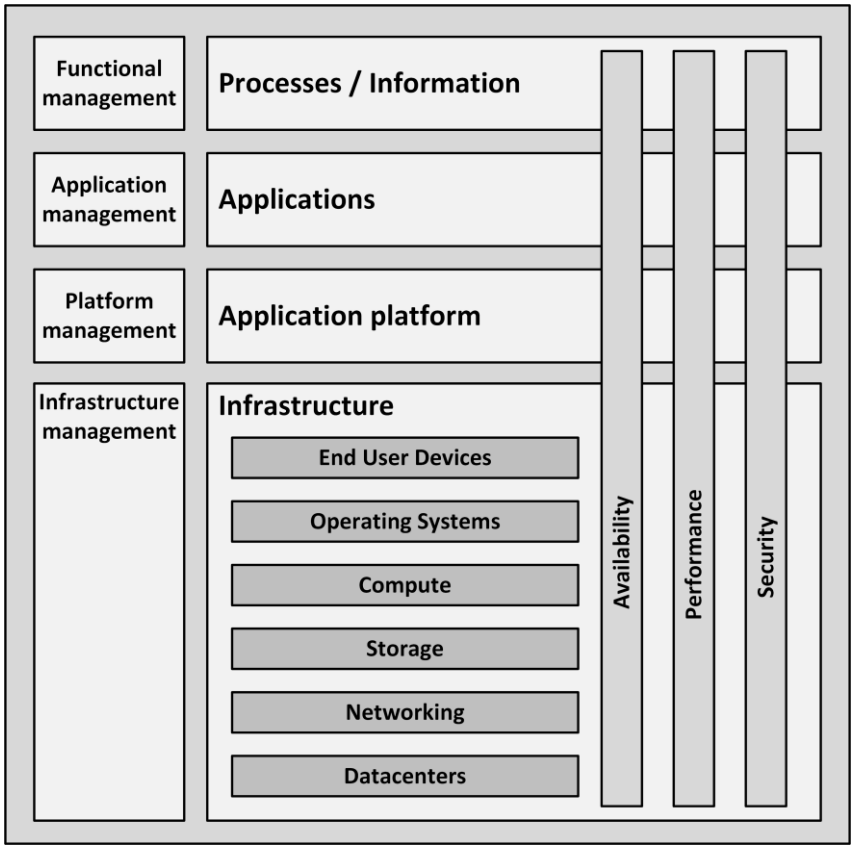


Figure 2: The infrastructure model

A model is always a simplified version of reality, useful to explain a certain point; not covering all details. Therefore, the infrastructure model is not perfect. As George E. P. Box once said: "*Essentially, all models are wrong, but some are useful.*"[1]

The following sections provide a high-level description of the building blocks in the infrastructure model.

2.2 Processes / Information building block

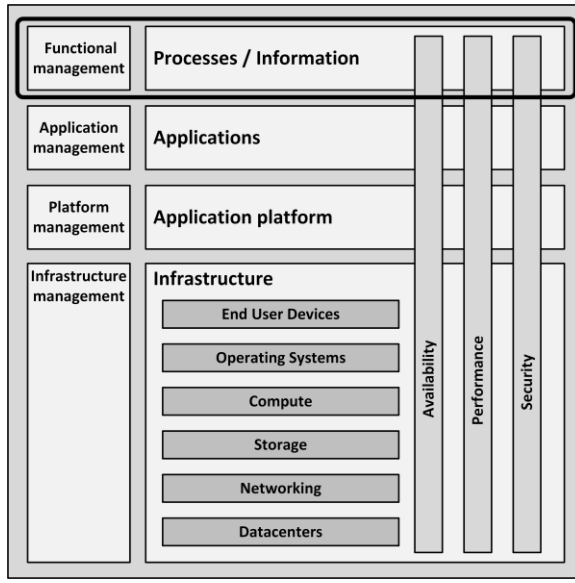


Figure 3: Processes / Information building block

Organizations implement business processes to fulfil their mission and vision. These processes are organization specific – they are the main differentiators between organizations. As an example, some business processes in an insurance company could be claim registration, claim payment, and create invoice.

Business processes create and use information. In our example, information could be the claim's date or the number of dollars on an invoice. Information is typically entered, stored and processed using applications.

Functional management is the category of systems management that ensures the system is configured to perform the required business functions.

2.3 Applications building block

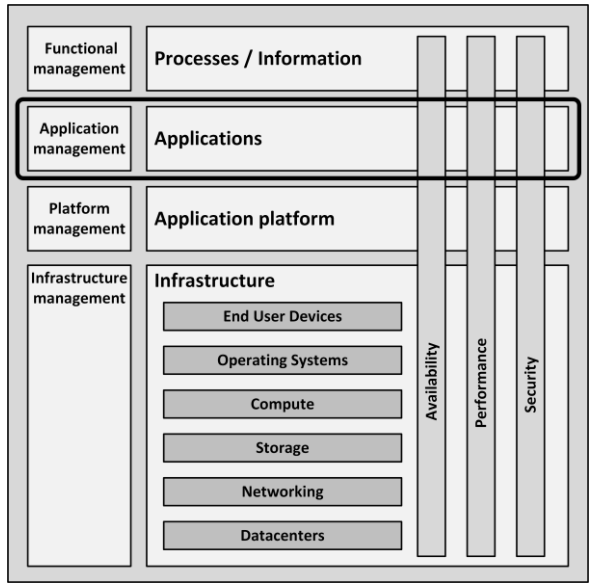


Figure 4: Applications building block

The Applications building block includes several types of applications based on the following characteristics:

- **Usage:** Applications can be single-user or multi-user. A single-user application typically runs on end-user devices such as PCs and laptops. Examples include web browsers, word processors, and email clients. Examples of multi-user applications include mail servers, portals, collaboration tools, and instant messaging servers.
- **Source:** Applications can be purchased as commercial off-the-shelf (COTS) products or developed as custom software.
- **Architecture:** Applications can be designed as standalone applications or as multi-tier applications. A multi-tier application consists of a number of layers, such as a JavaScript application in a browser that communicates with an on-premises web server, which communicates with an application server, which communicates with a database.
- **Timeliness:** Interactive applications respond to user actions, such as mouse clicks. They typically respond in the range of 100 to 300 ms. Real-time systems, such as Supervisory Control And Data

Acquisition (SCADA) systems, are used in manufacturing, logistics, or other environments where timeliness is critical. These systems must respond in less than 10 ms. At the other end of the spectrum are batch-based systems that process data for hours at a time.

Each of these types of applications requires a different type of underlying infrastructure.

Applications management is responsible for the configuration and technical operations of the applications.

2.4 Application Platform building block

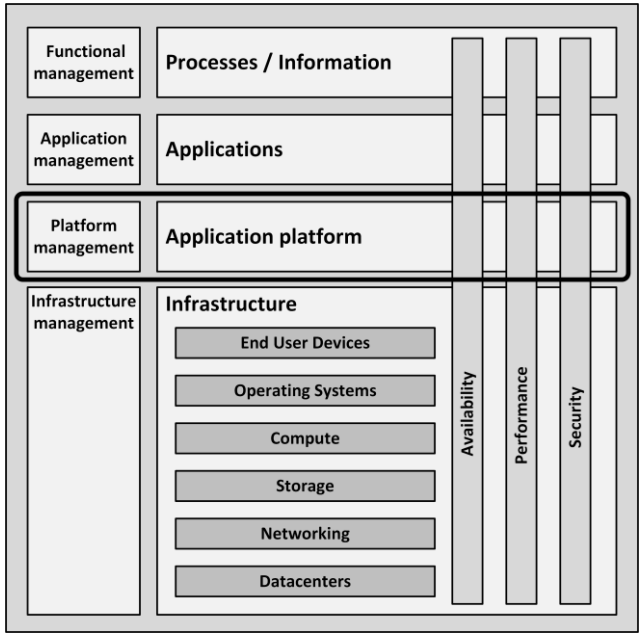


Figure 5: Application Platform building block

Most applications need some additional services, known as application platforms, that enable them to work. We can identify the following services as part of the application platform building block:

- **Application servers** provide services to applications. Examples are Java or .Net application servers and frameworks like IBM WebSphere, Apache Tomcat, and Red Hat JBoss.

- **Container platforms** like Kubernetes, Azure Container Instances, and Amazon Elastic Container Service, that run docker containers.
- **Connectivity** entails Enterprise Service Buses (ESBs) like Microsoft BizTalk, the TIBCO Service Bus, IBM MQ, and SAP NetWeaver PI.
- **Databases,** also known as database management systems (DBMSs), provide a way to store and retrieve structured data. Examples are Oracle RDBMS, IBM DB2, Microsoft SQL Server, PostgreSQL, MySQL, Apache CouchDB, and MongoDB.

Application platforms are typically managed by systems managers specialized in the specific technology.

2.5 Infrastructure building blocks

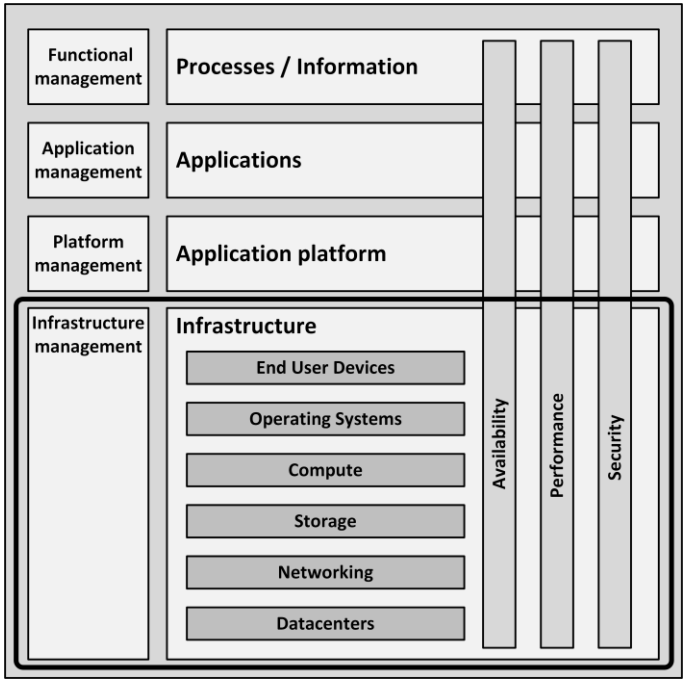


Figure 6: Infrastructure building block

This book uses the selection of building blocks as depicted in Figure 6 to describe the infrastructure building blocks and concepts – the scope of this book.

The following infrastructure building blocks are in scope:

- **End User Devices** are the devices used by end users to work with applications, like PCs, laptops, thin clients, mobile devices, and printers.
- **Operating Systems** are collections of programs that manage a computer's internal workings: its memory, processors, devices, and file system.
- **Compute** are the physical and virtual computers in the datacenter, also known as servers.
- **Storage** are systems that store data. They include hard disks, tapes, Direct Attached Storage (DAS), Network Attached Storage (NAS), and Storage Area Networks (SANs).
- **Networking** is used to connect all infrastructure components. This building block includes routers, switches, firewalls, WANs (wide area networks), local area networks (LANs), internet access, and VPNs (Virtual Private Network), and (on the network application level) networking services like DNS, DHCP, and time services, necessary for the infrastructure to work properly.
- **Datacenters** are locations that host most IT infrastructure hardware. They include facilities like uninterruptible power supplies (UPSs), Heating, Ventilation, and Air Conditioning (HVAC), computer racks, and physical security measures.

Please note that these building blocks are not per definition hierarchically related. For instance, servers need both networking and storage, and both are equally important.

Infrastructure management includes processes like ITIL and DevOps, and tools for monitoring, backup, and logging.

2.6 Non-Functional attributes

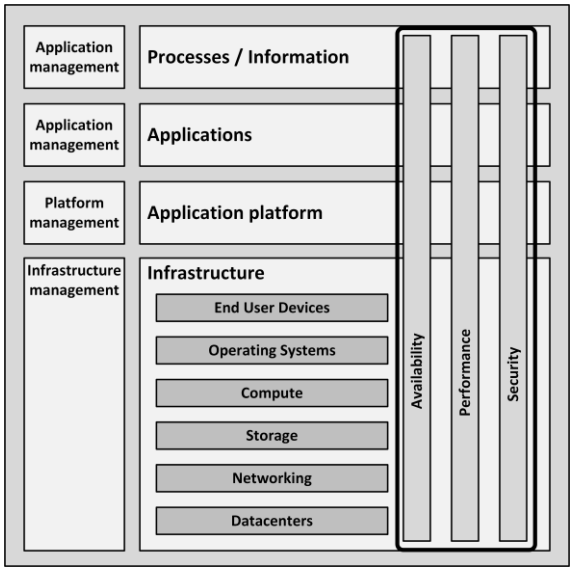


Figure 7: Non-Functional attributes

An IT system does not only provide functionality to users; functionality is supported by non-functional attributes. Non-functional attributes result from the configuration of all IT system components, both at the infrastructure level and above.

Although many other non-functional attributes are defined, as described in chapter 4, availability, performance, and security are almost always the essential ones in IT infrastructure architectures (Figure 7).

3

CLOUD COMPUTING AND INFRASTRUCTURES

In recent years, we have seen the widespread adoption of cloud computing. Cloud computing can be seen as one of the most important paradigm shifts in computing in recent years. Many organizations now have a cloud-first strategy and are taking steps to move applications from their own on-premises datacenters to the cloud managed by cloudproviders.

The term cloud is not new. In 1997, Ramnath Chellappa of the University of Texas already stated:

*Computing has evolved from a mainframe-based structure to a network-based architecture. While many terms have appeared to describe these new forms, the advent of electronic commerce has led to the emergence of '**cloud computing**'.*

While there are many public cloud service providers today, the three largest are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Together, these three have 66% of the market share and have a large number of datacenters around the world. Figure 8 shows when each of these cloud providers started.

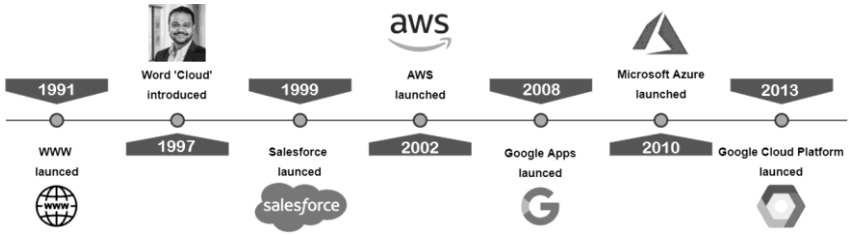


Figure 8: Cloud time line

The three major cloud providers offer similar services, but sometimes under different names. For instance, a virtual machine in Azure is just called a virtual machine, but in GCP it is called a Compute Engine and in AWS it is called an EC2 instance.

While cloud computing can be seen as the new infrastructure, many organizations will be using on-premises infrastructure for many years to come. Migrating a complex application landscape to a cloud provider is no simple task and can take years. And maybe an organization is not allowed to take all its applications to the cloud. In many cases, there will be a hybrid situation, with part of the infrastructure on-premises and another part in one or more clouds.

Please be aware that the cloud is just a number of datacenters that are still filled with hardware – compute, networking and storage. Therefore, it is good to understand infrastructure building blocks and principles even when moving to the cloud,

3.1 Cloud definition

The most accepted definition of cloud computing is that of the National Institute of Standards and Technology (NIST)[2]:

*Cloud computing is 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..*

It is important to realize that cloud computing is not about technology; it is an *outsourcing business model*. It enables organizations to cut cost while at the same time focusing on their primary business – they should focus on running their business instead of running a mail server.

Clouds are composed of five essential characteristics, four deployment models, and three service models.

3.2 Cloud characteristics

Essential cloud characteristics are:

- **On demand self-service** – As a result of optimal automation and orchestration, minimal systems management effort is needed to deploy systems or applications in a cloud environment. In most cases, end uses can configure, deploy, start and stop systems or applications on demand.
- **Rapid elasticity** – A cloud is able to quickly scale-up and scale-down resources. When temporarily more processing power or storage is needed, for instance as a result of a high-exposure business marketing campaign, a cloud can scale-up very quickly on demand. When demand decreases, cloud resources can rapidly scale down, leading to elasticity of resources.
- **Resource pooling** – Instead of providing each application with a fixed amount of processing power and storage, cloud computing provides applications with resources from a shared pool. This is typically implemented using virtualization technologies.
- **Measured service** – In a cloud environment the actual resource usage is measured and billed. There are no capital expenses, only operational expenses. This in contrast with the investments needed to build a traditional infrastructure.
- **Broad network access** – Capabilities are available over the network and accessed through standard mechanisms.

Be aware that when using public cloud based solutions, the internet connection becomes a Single Point of Failure. Internet availability and internet performance becomes critical and redundant connectivity is therefore key.

3.3 Cloud deployment models

A cloud can be implemented in one of four deployment models.

- A **public cloud** deployment is delivered by a cloud service provider, is accessible through the internet, and available to the general public. Because of their large customer base, public clouds largely benefit from economies of scale.

- A **private cloud** is operated solely for a single organization, whether managed internally or by a third-party, and hosted either on premises or external. It extensively uses virtualization and standardization to bring down systems management cost and staff.
- A **community cloud** is much like a private cloud, but shared with a community of organizations that have shared concerns (like compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination, and it may exist on or off premises.
- In a **hybrid cloud** deployment, a service or application is provided by a combination of a public cloud, and a community cloud and/or a private cloud. This enables running generic services (like email servers) in the public cloud while hosting specialized services (like a business specific application) in the private or community cloud.

3.4 Cloud service models

Clouds can be delivered in one of three service models:

- **Software-as-a-Service (SaaS)** delivers full applications that can be used by business users, and need little or no configuration. Examples are Microsoft Office365, LinkedIn, Facebook, Twitter, and Salesforce.com.
- **Platform-as-a-Service (PaaS)** delivers a scalable, high available, open programming platform that can be used by developers to build bespoke applications that run on the PaaS platform. Examples are Microsoft Azure Cloud Service and Google App Engine.
- **Infrastructure-as-a-Service (IaaS)** delivers (virtual) machines, networking, and storage. The user needs to install and maintain the operating systems and the layers above that. Examples are Amazon Elastic Cloud (EC2 and S3) and Microsoft Azure IaaS.

The following figure shows the responsibility of the cloud provider for each service model.

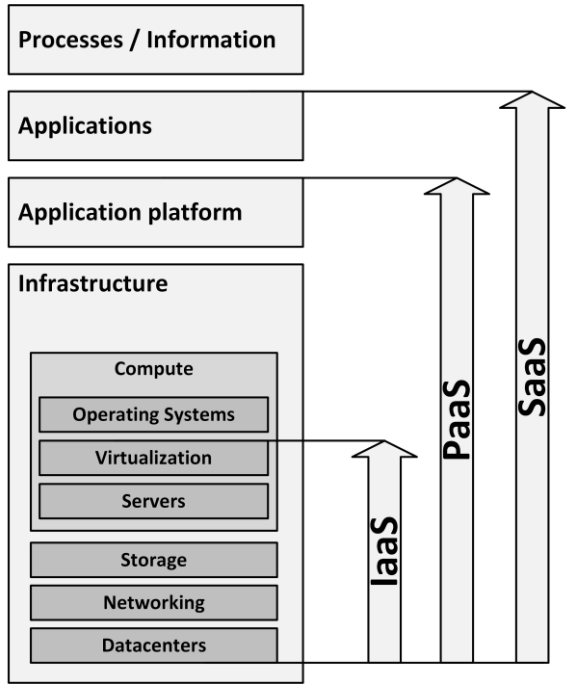


Figure 9: Cloud provider responsibilities

In the context of this book, IaaS is the most relevant service model.

When we combine both deployment and service models, we get the following picture.

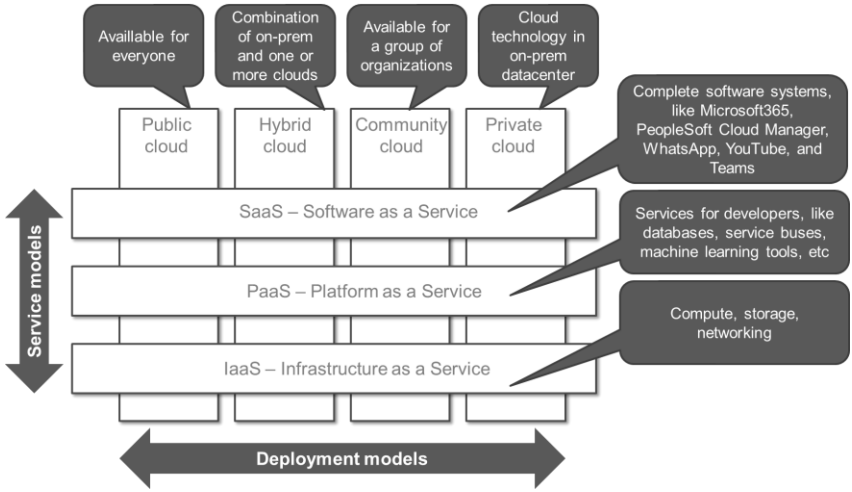


Figure 10: Cloud models

Because of the scope of this book, the next section describes Infrastructure as s Service in more detail.

3.5 Infrastructure as a Service (IaaS)

Infrastructure as a Service provides virtual machines, virtualized storage, virtualized networking and the systems management tools to manage them. IaaS can be configured using a graphical user interface (GUI), a command line interface (CLI), or application programming interfaces (APIs).

IaaS is typically based on cheap commodity white label hardware. The philosophy is to keep the cost down by allowing the hardware to fail every now and then. Failed components are either replaced or simply removed from the pool of available resources.

IaaS provides simple, highly standardized building blocks to applications. It does not provide high availability, guaranteed performance or extensive security controls. Consequently, applications running on IaaS should be robust to allow for failing hardware and should be horizontally scalable to increase performance.

In order to use IaaS, users must create and start a new server, and then install an operating system and their applications. Since the cloud provider only provides basic services, like billing and monitoring, the user is responsible for patching and maintaining the operating systems and application software.

Not all operating systems and applications can be used in an IaaS cloud; some software licenses prohibit the use of a fully scalable, virtual environment like IaaS, where it is impossible to know in advance on which machines software will run.

3.6 Edge computing

The goal of edge computing is to bring computing power and data storage closer to where it is needed, rather than relying on a cloud or on-premises datacenter. In edge computing, compute and storage take place on devices at the edge of the network, such as routers, gateways, switches, and sensors.

Edge computing can be a viable option where low latency, high bandwidth, and real-time processing are critical. For example, in the case of autonomous vehicles, real-time decision making is critical for safety. In this scenario, edge computing can enable the vehicle to process data and make decisions locally, rather than sending all sensor data to a centralized datacenter.

Edge computing is also gaining popularity in Internet of Things (IoT) applications, where a large number of devices generate data that must be processed in real time. By using edge computing, organizations can reduce the amount of data that needs to be sent to the cloud, which can reduce costs and improve performance.

PART II
–
NON FUNCTONAL ATTRIBUTES

It's hardware that makes a machine fast. It's software that makes a fast machine slow.

Craig Bruce

4

INTRODUCTION TO NON-FUNCTIONAL ATTRIBUTES

4.1 Introduction

IT infrastructures provide services to applications. Some of these infrastructure services can be well defined, like providing disk space, or routing network messages. Non-functional attributes, on the other hand, describe the qualitative behavior of a system, rather than specific functionalities. Some examples of non-functional attributes are scalability, reliability, stability, testability, and recoverability. But in my experience, the three most important non-functional attributes for IT infrastructures are **security**, **performance**, and **availability**. Therefore, for each topic described in this book, these three non-functionals attributes are explicitly addressed.

Non-functional attributes are very important for the successful implementation and use of an IT infrastructure, but in projects, they rarely get the same attention as the functional services.

Not everybody is aware of the value of pursuing non-functional attributes. The name suggests they have no function. But of course, these attributes do have a function in the business process, and usually a fairly large one. For instance, when the infrastructure of a corporate website is not performing well, the visitors of the website will leave, which has a direct financial impact on the business. When credit card transactions are not stored in a secure way in the infrastructure, and as a result leak to hackers, the organization that stored the credit card data will have a lot of explaining to do to their customers.

So, non-functional attributes are very functional indeed, but they are not directly related to the primary functionalities of a system. Instead of the term non-functional requirement, it would be much better to use the term quality attributes or implicit requirements. Although these terms much better reflect the nature and importance of, for example, performance, security, and availability, the term non-functional requirement (as expressed in non-functional requirements or NFRs) is more commonly used and widely known. Therefore, in this book I keep on using the term non-functional attribute, although I do realize that the term could be misleading.

While architects and certainly infrastructure specialists are usually very aware of the importance of non-functional attributes of their infrastructure, many other stakeholders may not have the same feelings about it. Users normally think of functionalities, while non-functional attributes are considered a hygiene factor and taken for granted ("*Of course, the system must perform well*"). Users of systems most of the time don't state non-functional attributes explicitly, but they do have expectations about them.

An example is the functionality of a car.

A car has to bring you from A to B, but many quality attributes are taken for granted.

For instance, the car has to be safe to drive in (leading to the implementation of anti-lock brakes, air bags, and safety belts) and reliable (the car should not break down every day), and the car must adhere to certain industry standards (the gas pedal must be the right-most pedal).

All of these extras cost money and might complicate the design, construction, and maintenance of the car. While all clients have these non-functional requirements, they are almost never expressed as such when people are ordering a new car.

4.2 Non-functional Requirements

It is the IT architect or requirements engineer's job to find implicit requirements on non-functional requirements. This can be very hard, since what is obvious or taken for granted by customers or end users of a system is not always obvious to the designers and builders of that system. Not to mention the non-functional requirements of other stakeholders, such as the existence of service windows or monitoring capabilities, which are important requirements for systems managers.

SKIPPED TEXT

4.2.1 MTBF and MTTR

The factors involved in calculating availability are Mean Time Between Failures (MTBF), which is the average time that passes between failures, and Mean Time To Repair (MTTR), also known as Mean Time To Recover, which is the time it takes to recover from a failure.

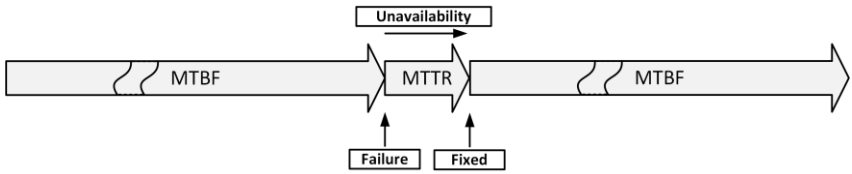


Figure 12: MTBF and MTTR

The term "mean" means that the numbers expressed by MTBF and MTTR are statistically calculated values.

4.2.1.1 Mean Time Between Failures (MTBF)

The MTBF is expressed in hours (how many hours will the component or service work without failure). Some typical MTBF figures are shown in Table 3.

Component	MTBF (hours)
Hard disk	750,000
Power supply	100,000
Fan	100,000
Ethernet Network Switch	350,000
RAM	1,000,000

Table 3: MTBF levels

It is important to understand how these numbers are calculated. No manufacturer can test if a hard disk will continue to work without failing for 750,000 hours (= 85 years). Instead, manufacturers run tests on large batches of components. In case of for instance hard disks, 1000 disks van be tested for 3 months. If in that period of time five disks fail, the MTBF is calculated as follows:

The test time is 3 months. One year has four of those periods. So, if the test would have lasted one year, 4 × 5 = 20 disks would have failed.

In one year, the disks would have run in total:

1000 disks × 365 × 24 = 8,760,000 running hours.

This means that the MTBF $= \frac{8{,}760{,}000\ hours}{20\ failed\ drives} = 438{,}000$ hours/failure.

So, actually MTBF only says something about the chance of failure in the first months of use. It is an extrapolated value for the probable downtime of a disk. It would be better to specify the annual failure rate instead (in our example, 2% of all disks will fail in the first year), but that is not very good advertising.

4.2.1.2 Mean Time To Repair (MTTR)

When a component breaks, it needs to be repaired. Usually the repair time (expressed as Mean Time To Repair – MTTR) is kept low by having a service contract with the supplier of the component. Sometimes spare parts are kept on-site to lower the MTTR (making MTTR more like Mean Time To Replace). Typically, a faulty component is not repaired immediately. Some examples of what might be needed for to complete repairs are:

- Notification of the fault (time before seeing an alarm message)
- Processing the alarm
- Finding the root cause of the error
- Looking up repair information
- Getting spare components from storage
- Having technician come to the datacenter with the spare component
- Physically repairing the fault
- Restarting and testing the component

Instead of these manual actions, the best way to keep the MTTR low is to introduce automated redundancy and failover, as discussed in sections 5.4.1 and 5.4.2.

4.2.2 Some calculation examples

Decreasing MTTR and increasing MTBF both increase availability. Dividing MTBF by the sum of MTBF and MTTR results in the availability expressed as a percentage: Availability $= \frac{\text{MTBF}}{(\text{MTBF+MTTR})} \times 100\%$.

For example:

A power supply's MTBF is 150,000 hours. This means that on average this power supply fails once every 150,000 hours (= once per 17 years). If the time to repair the power supply is 8 hours, the availability can be calculated as follows: $\text{Availability} = \frac{150{,}000 \text{ hours}}{(150{,}000 \text{ hours} + 8 \text{ hours})} \times 100\% = 99.99466\ \%$

This means that because of the repair time alone this component can never reach an average availability of 99.999%! To reach five nines of availability the repair time should be as low as 90 minutes for this component. Note that if a downtime of 99.999% is acceptable per year (and not over the total lifetime of the component), the repair time must be lower than 6 minutes!

As system complexity increases, usually availability decreases. When a failure of any one part in a system causes a failure of the system as a whole, the availability is called serial availability. To calculate the availability of such a complex system or device, multiply the availability of all its parts.

For example, a server consists of the following components and the MTTR of any part of the server is 8 hours.

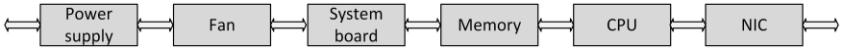


Figure 13: System with serial components

Component	MTBF (h)	MTTR (h)	Availability	in %
Power supply	100,000	8	0.9999200	99.99200
Fan	100,000	8	0.9999200	99.99200
System board	300,000	8	0.9999733	99.99733
Memory	1,000,000	8	0,9999920	99.99920
CPU	500,000	8	0.9999840	99.99840
Network Interface Controller (NIC)	250,000	8	0.9999680	99.99680

Table 4: Availability in percentages

The availability of the total server is: $0.9999200 \times 0.9999200 \times 0.9999733 \times 0.9999920 \times 0.9999840 \times 0.9999680 = 0.9997733 = 99.977\%$. This is lower than the availability of any single component in the system. Therefore, the more components a system includes (and each component is critical for the total system), the lower the total availability becomes.

To increase the availability, systems (composed of a various components) can be deployed in parallel. This considerably increases the availability, since the

combined system no longer contains a Single Point Of Failure. If one component becomes unavailable, the affected system goes down, but the other system can take over. Consider the example below. Two systems run in parallel, each complete system having an availability of 99%.

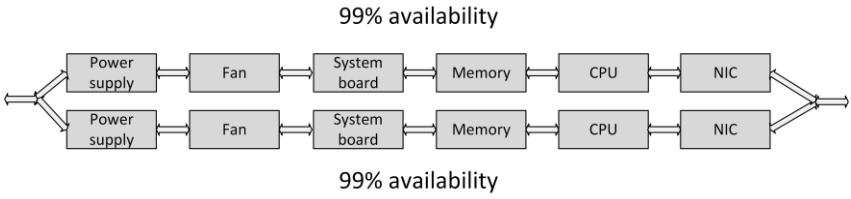


Figure 14: Two systems in parallel

The chance of both systems being unavailable at the same time is very small and can be calculated as follows[3]:

$$A_T = 1 - \prod_{i=1}^{n} (1 - A_i)$$

where

A_T is the total availability of the configuration

n is the total number of systems in parallel

A_i is the availability of the i-th system

As an example, let's assume an organization uses two internet connections, each from another provider. The first is their primary connection, which has an average uptime of 99.99%. The second connection is the backup connection in case the first one fails. This one has an average uptime of 99.9%. This leads to a combined uptime of:

$1 - (1 - 0.9999) \times (1 - 0.999) = 0.9999999$ (or 99.99999%), which is significantly higher than each of the individual connections.

Another example: when the availability for each system is estimated to be 99%, the combined availability in a parallel setup is:

Situation	Availability	Yearly downtime
1 system	99%	87h 36m
2 systems	99.99%	52m
3 systems	99.9999%	32s

Table 5: Availability with multiple components

In this situation, it is important to have no single point of failure that combines the set of systems (for instance, all systems run on the same power supply). In that case, the availability of the system is fully dependent on that one component.

It is often not useful to have more than two systems to achieve high availability. Studies[4] show that having dual redundancy for a system is sufficient because other components of the system, such as human error from system administrators, have orders of magnitude worse MTBF. Therefore, adding a (costly and complex) third system would be cancelled out by unavailability due to other causes. Triple redundancy would not increase the overall availability of the system, but actually decrease it, due to the complexity of its application.

SKIPPED TEXT

4.2.3 Scalable cloud environments

In cloud environments, there is often no need to guess the required capacity. Since resources such as virtual machines are easily scalable in cloud environments, you can simply start with a large system and scale down if the system is underutilized. Or vice versa: Start with a small system and scale up until the performance is acceptable.

Of course, this only works well if you can reliably measure system performance. Fortunately, cloud environments usually have very extensive logging and monitoring capabilities to help with this.

4.3 Performance of a running system

4.3.1 Managing bottlenecks

The performance of a system is based on the performance of all its components, and the interoperability of various components. Therefore, measuring the performance of a system only has value if the complete system is taken into

account. For instance, building an infrastructure with really fast networking components has little benefits when the used hard disks are slow.

A performance problem may be identified by slow or unresponsive systems. This usually occurs because of high system loads, causing some component of the system to reach some limit. This component is referred to as the bottleneck of the system, because the performance or capacity of the entire system is limited by a single component, slowing down the system as a whole. To find this bottleneck, performance measurements are needed.

Only when we know where in the system the bottleneck occurs, we can try to improve performance by removing that bottleneck.

When a bottleneck is removed, usually another bottleneck arises. In fact, no matter how much performance tuning is done, there will always be a bottleneck somewhere. According to the Bottleneck law[5], every system, regardless of how well it works, has at least one constraint (a bottleneck) that limits its performance. This is perfectly fine when the bottleneck does not negatively affect the performance of the entire system to the point where the stated performance requirements are no longer met.

Benchmarking is a way to measure individual components, while system performance tests measure the system as a whole.

4.3.2 Performance testing

There are three major types of performance tests for testing complete systems:

- **Load testing** - This test shows how a system performs under the expected load. It is a check to see if the system performs well under normal circumstances.
- **Stress testing** - This test shows how a system reacts when it is under extreme load. Goal is to see at what point the system "breaks" (the breakpoint, as shown in Figure 18) and *where* it breaks (the bottleneck).
- **Endurance testing** - This test shows how a system behaves when it is used at the expected load for a long period of time. Typical issues that arise are memory leaks, expanding database tables, or filling up disks, leading to performance degradation.

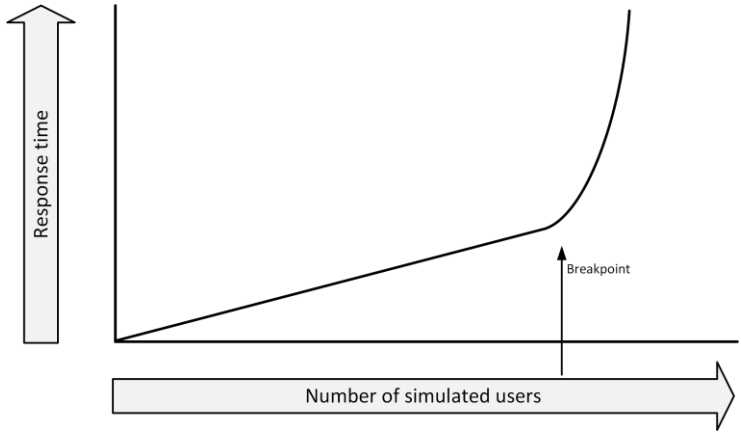


Figure 18: Performance breakpoint

Performance testing software typically uses one or more servers to act as injectors – each emulating a number of users that run a sequence of interactions (recorded as a script, or as a series of scripts to emulate different types of user interaction). A separate server acts as a test conductor, coordinating the tasks, gathering metrics from each of the injectors, and collecting performance data for reporting purposes.

The usual sequence is to ramp up the load – starting with a small number of virtual users and increasing the number over a period of time to some maximum. The test result shows how the performance varies with the load, given as number of users versus response time.

A cloud environment is ideal for setting up performance testing environments because it can be scaled up to provide the required load and then scaled down after the test is complete. This can reduce the cost of running a performance test while simulating a very large number of users during the test period.

Performance testing should be done in a production-like environment. Performance testing in a development environment usually produces results that have little meaning for what is likely to happen in production. To reduce costs, it is sometimes advisable to use a temporary test environment, such as one rented from your hardware vendor that has the same components as the production environment. If the test environment is underpowered (the machines are not as fast as production, the disks are of a different type, etc.), the test results cannot be relied upon because they are not comparable to the production environment. Even if the underpowered test systems perform well enough to get good test results, the faster production system may have performance issues that were not experienced in the tests.

I have experienced such a situation: A production system was much faster than the test system we used. While the tests showed no performance issues on the slower test system, the application performed badly on the faster production systems.

The reason was a network protocol that could not receive network packages as fast as the production systems could provide it.

4.4 Performance patterns

There are various ways to improve the performance of systems. This section describes caching, scaling, load balancing, high performance computing, designing for performance, and capacity management.

But first a quick word on increasing performance on other levels than the infrastructure.

4.4.1 Increasing performance on upper layers

Experience learns that 80% of the performance issues are due to badly behaving applications. While much effort can be put in optimizing infrastructure performance, it is good practice to first check for performance optimizations in the upper layers. Database and application tuning typically provides much more opportunity for performance increase than installing more computing power.

I have seen a management report that used to run for 45 minutes. After tuning the database, it ran in 3 minutes, just by optimizing some SQL queries and adding a database index. Increasing the performance that much in the infrastructure layer instead is not only very complicated but also very expensive!

Another example was a badly programmed application where each read and write to disk opened and closed the file, instead of opening the file at the start of the application and keeping it open until the application is stopped.

Since opening and closing files is much slower than the actual reading or writing of data, just keeping files open vastly increased the performance of the application.

Application performance can benefit from prioritizing tasks, working from memory as much as possible (as opposed to working with data on disk), and making good use of queues and schedulers.

Of course, bad behaving applications can only be fixed when you have access to the application's source code. For commercial off-the-shelf software, this is usually not feasible. Tuning the databases used by the application, by for instance adding indexes, can be an opportunity to significantly improve performance. Fortunately, today's databases use automated query optimizing, where the performance of often used queries automatically gets better over time.

In the current era of multi-core processors, it is important for application developers to understand how applications work on a multithreaded system. Unfortunately, this is not always the case and many applications run on only one of the available cores of the CPU.

Intel introduced circuitry in its processors that can boost the clock speed of one of the cores when a running single threaded application is detected. This boost of the clock speed would normally introduce too much heat in the processor, but since the other cores are not performing any work in a single threaded application, the overall temperature of the CPU stays within range.

4.4.2 Caching

Caching improves performance by retaining frequently used data in high-speed memory, reducing access times to data.

Some sources that provide data are slower than others. The approximate speed of retrieving data from various sources is shown in Table 8.

Component	**Time it takes to fetch *1 MB* of data (ms)**
Network, 1 Gbit/s	675
Hard disk, 15k rpm, 4 KB disk blocks[6]	105
Main memory DDR3 RAM[7]	0.2
CPU L1 cache[8]	0.016

Table 8: Approximate speeds of fetching data

Especially in situations where retrieving data takes relatively long (for instance reading from hard disk or from the network), caching in memory can significantly improve performance.

4.4.2.1 Disk caching

Disks are mechanical devices that are relatively slow by nature. To speed up the reading of data from disk, disk drives contain cache memory. This cache memory stores all data recently read from disk, and some of the disk blocks following the recently read disk blocks. When the data is read again, or (more likely) the data of the following disk block is needed, it is fetched from high-speed cache memory.

Disk caching can be implemented in the storage component itself (for instance cache used on the physical disks or cache implemented in the disk controller), but also in the operating system. The general rule of thumb that adding memory in servers improves performance is due to the fact that all non-used memory in operating systems is used for disk cache. Over time, all memory gets filled with previously stored disk requests and prefetched disk blocks, speeding up applications.

4.4.3 Web proxies

Another example of caching is the use of web proxies. When users browse the internet, instead of fetching all requested data from the internet each time, earlier accessed data can be cached in a proxy server and fetched from there. This has two benefits: users get their data faster than when it would be retrieved from a distant web server, and all other users are provided more bandwidth to the internet, as the data did not have to be downloaded again.

4.4.4 Operational data store

An Operational Data Store (ODS) is a read-only replica of a part of a database for a specific use. Instead of accessing the main database for retrieving information, often used information is retrieved from a separate small ODS database, not degrading the performance of the main database.

A good example of this is a website of a bank. Most users want to see their actual balance when they login (and maybe the last 10 mutations of their balance). When every balance change is not only stored in the main database of the bank, but also in a small ODS database, the website only needs to access the ODS to provide users with the data they most likely need. This not only speeds up the user experience, but also decreases the load on the main database.

4.4.5 Front-end servers

In web facing environments storing most accessed (parts of) pages on the web front-end server (like the static pictures used on the landing page) significantly lowers the amount of traffic to back-end systems. Reverse-proxies can be used to automatically cache most requested data as well.

4.4.6 In-memory databases

In special circumstances, entire databases can be run from memory instead of from disk. These so-called in-memory databases are used in situations where performance is crucial, like in real-time SCADA systems and in high performance online transaction processing (OLTP) systems. Of course, special arrangements must be made to ensure data is not lost when a power failure occurs.

As an example, SAP HANA is an in-memory database for SAP enterprise resource planning (ERP) systems.

4.4.7 Edge servers

The major cloud providers have datacenters around the world. In addition, they often offer edge locations. These edge locations can be used to cache data in close proximity to end users.

For example, a Web site that offers streaming videos can place copies of those videos in a number of edge locations around the world to ensure a good user experience for users in all locations.

SKIPPED TEXT

4.5 Cloud security

The public cloud follows a shared responsibility model. In this model, the cloud provider takes care of security *of* the cloud, and the customer takes care of security *in* the cloud. Table 9 shows this sharing of responsibilities. Depending on the cloud deployment model used, the separation of responsibilities differs.

	On-premises	IaaS	PaaS	SaaS
Data classification	Client	Client	Client	Client
Application configuration	Client	Client	Client	Client
Identity & Access	Client	Client	Client	Client
Application	Client	Client	Client	Cloud
Operating system	Client	Client	Cloud	Cloud
Compute	Client	Cloud	Cloud	Cloud
Storage	Client	Cloud	Cloud	Cloud
Network	Client	Cloud	Cloud	Cloud
Physical security	Client	Cloud	Cloud	Cloud

Table 9: Shared responsibility model: Client versus Cloud provider

Although some people express doubts about the security of the public cloud, in practice it turns out that the public cloud is very secure. All components in the public cloud are designed with security in mind. A lot is invested in security by the cloud providers and a large number of specialists work daily to optimize cloud security. Few organizations can achieve such a high level of security in their own datacenter. So, the main question is: Do you really think you can do better yourself?

So, while the cloud is very secure, it can easily become insecure through incorrect use or configuration errors. An example is a virtual machine in Azure. These are given a public IP address by default. This makes the VM easy to find from the internet and gives hackers a chance to attack it. Therefore, when creating a new VM, either the public IP address must be disabled, or access to the IP address must be blocked with a firewall. By the way, a VM in AWS does not have a public IP address by default.

4.6 Security Patterns

Information can be stolen in many ways. Here are some of the more common ways related to infrastructure that you should be aware of:

- **Key loggers** can be maliciously installed on end user devices. They can send sensitive information like passwords to third parties.
- **Network sniffers** can show network packages that contain sensitive information or replay a logon sequence by which a hacker can successfully authenticate to an IT system.
- Data on **backup tapes** outside of the building can get into wrong hands.

- PCs or disks that are **disposed** of can get into the wrong hands.
- Corrupt or dissatisfied **staff** can copy information.
- **End users** are led to a malicious website that steals information (also known as phishing).

To deal with these risks, proper security processes must be put in place, based on the prevention of security problems, the detection of security breaches as soon as they occur, and the responsive actions to be taken to minimize damage and restore operations.

4.6.1 Prevention

4.6.1.1 Security policies

Managing security is all about managing risks. If there are no risks, we don't need any security controls. The effort we put in securing the infrastructure should therefore be directly related to the risk at hand. Risk can be mitigated. Here are some techniques to make data secure:

- **Design for minimum risk**. Design the system to eliminate as much vulnerabilities as possible. This can for instance be done using source code analysis in software development and by running critical systems stand-alone instead of connected to other systems.
- **Incorporate safety devices**. Reduce risk using devices like firewalls and hardened screened routers. These devices usually don't affect the probability, but reduce the severity of an exploit: an automobile seat belt doesn't prevent a collision, but reduces the severity of injuries. A firewall does not prevent attacks, but reduces the chance of an attacker connecting to sensitive parts of the network.
- **Implement training and procedures.** These can mitigate risks that are people-bound like social engineering attacks.

Ensure that appropriate security processes are in place and regularly tested. It should be clear who is responsible for security monitoring, what should happen if a security incident is detected, and what security processes should be followed before new or modified software is put into production.

In addition, it is often possible to have security policies monitored automatically. Cloud providers in particular often provide options to enforce and monitor policies for all cloud resources in use. There are sets of predefined policies available, such as the CIS benchmarks[9]. These are general best practices, for operating systems, network components, desktops, virtualization platforms and cloud services, among others. CIS benchmarks can be

implemented and monitored manually, but they can also be enforced automatically in, for example, a public cloud environment.

4.6.1.2 Zero Trust

Zero trust is a security framework that moves defense from network perimeters to a combination of users, resources and locations (who wants to use what and where). Zero trust does not trust people and systems based on their location, but requires all users, inside or outside the organization's network, to be continuously authenticated, authorized and validated before being granted access to applications and data.

Zero Trust assumes that there is no traditional network edge; networks can be local, in the cloud, or hybrid with users in any location.

A zero trust architecture includes continuous multi-factor authentication, network micro-segmentation, encryption, endpoint security, analytics and robust auditing.

Zero trust became popular as a response to hybrid cloud environments, where data is stored both on-premises and in public clouds and where users must be able to access the data securely from any location.

4.6.1.3 Segregation of duties and least privilege

Segregation of duties (also known as separation of duties) assigns related sensitive tasks to different people or departments. The reasoning is that if no single person has total control of the system’s security mechanisms, no single person can compromise the system.

This concept is related to the principle of least privilege. Least privilege means that users of a system should have the lowest level of privileges necessary to perform their work, and should only have them for the shortest length of time.

In many organizations, a systems manager has full control over the system’s administration and security functions. In general, this is a bad idea. Security tasks should not automatically be given to the systems manager. In secure systems, multiple distinct administrative roles should be configured, like a security manager, a systems manager, and a super user.

The security administrator, the system administrator and the super user do not necessarily have to be different people (but that is of course preferable). But when, for example, a system administrator takes on the role of the security administrator, this role change is monitored, logged and audited. Although it may be inconvenient for the person to switch from one role to another, the roles are functionally distinct and must be performed as such to maintain a high level of security.

In addition, a two-man control policy can be applied, in which two systems managers must review and approve each other's work. The purpose of two-man control (also known as the four eyes principle) is to minimize fraud or mistakes in highly sensitive or high-risk transactions. With two-man control, two systems managers are needed to complete every security sensitive task.

4.6.1.4 Privileged Access Management (PAM)

Privileged Access Management (PAM) is a network component that provides secure access to systems or parts of systems (such as databases) by users who require high privileges.

An example is the administrator or root account of an operating system. Under normal circumstances, this account is never used for day-to-day tasks, even by systems administrators. In exceptional cases, however, these accounts may be needed. In this case, the systems manager does not log into the operating system directly, but logs into a PAM system with his own credentials, using multi-factor authentication. The PAM has a password vault and uses the administrator or root password to log into the required operating system on behalf of the user. The user can then perform the intended work. When finished, the user logs out of the PAM and the PAM immediately changes the administrator or root password. This way, the user will never know what the password was or what the new password is.

All actions on a PAM system are logged for auditing purposes. As an additional feature, many PAM systems can log all keystrokes to audit logs, and it is sometimes possible to automatically capture a video of the actions on the screen.

4.6.1.5 Layered security

A layered security strategy is a good practice to enhance the overall IT security. The essence of layered security (also known as a Defense-In-Depth strategy) is to implement security measures in various parts of the IT infrastructure. This approach is comparable with physical security.

If a burglar wants to steal money from your house, he has to climb over the fence in the garden, then he has to get through a closed front door with locks, then he has to find the safe with the money, he has to break into the safe, get the money, and leave the premises. All of this must be done without being seen or heard; he must not be noticed by anyone during all of these steps.

It is obvious why this layered security works so well:

- Many barriers must be crossed (fence, door, safe).

- Opening every barrier takes different technical skills (climbing over the fence, lock picking a door with a mechanical lock, opening a safe with a digital lock).

- The burglar is slowed down by every barrier he tempts to cross, which increases the possibility of detection.

- The burglar doesn't know in advance how many barriers he has to cross, how much time each barrier takes, and which knowledge is needed for every barrier.

- The chance of getting caught is present in every step.

- When one barrier is crossed, the security of all other barriers is still intact.

In IT infrastructure, instead of having one big firewall and have all your security depend on it, it is better to implement several layers of security. Preferably these layers make use of different technologies, which makes it harder for hackers to break through all barriers; they will need a lot of knowledge for each step.

Each layer can be integrated with an Intrusion Detection System (IDS – see section 7.5.2.2) or some other system that detects break-ins, which increases the chance of getting caught. On top of this, more layers introduce uncertainty for the hacker: it is unknown many barriers must be passed to get to the data, and how long will this take, leading to demotivation. And if one layer is passed unnoticed, or if one security layer contains a vulnerability, the total security is still intact, albeit with less layers.

A disadvantage of implementing layered security is that it increases the complexity of the system. Every security layer must be managed, and systems managers must have knowledge about all used technologies.

SKIPPED TEXT

Picture 4: Computer racks[10]

More flexible air cooling replaced the traditional water cooling and sophisticated fire prevention, detection and extinguishing systems were installed. Because almost all work on the servers could now be done without touching the physical equipment, lights-out datacenters were introduced, where during normal operations no people are needed inside the datacenter, and the lights could thus be switched off.

The pace of innovation in datacenters is increasing, driven by cloud service providers and large-scale datacenters running internet applications like search engines, video streaming, and social media.

Very large datacenters today contain shipping containers packed with thousands of servers each. When repairs or upgrades are needed, entire containers are replaced (rather than repairing individual servers).

4.7 Datacenter building blocks

4.7.1 Datacenter categories

A datacenter can occupy one room in a building, one or more floors, or an entire building. Below are four typical datacenter categories.

- **Sub Equipment Room (SER)** – a SER is also known as a patch closet. They contain patch panels for connections to wall outlets in offices and some small equipment like network switches.

- **Main Equipment Room (MER)** – a MER is a small datacenter in the organization's subsidiaries or buildings.
- **Organization owned datacenter** – a datacenter that contains all central IT equipment for the organization. An organization can have multiple datacenters, often with failover and fallback capabilities.
- **Multi-tenant datacenter (also known as co-location)** – this datacenter category is owned by service providers that provide services for multiple other organizations.

If the datacenter is used for one organization only, it makes sense to install the datacenter inside one of the office buildings. But when the datacenter is used by multiple organizations, like in case of an internet service provider, choosing a location of the datacenter is more difficult.

4.7.2 Cloud datacenters

Cloud datacenters that are owned by large public cloud providers are also called hyperscaler datacenters, since they are hosting a very large number of servers, networking equipment and storage systems. Datacenters of large public cloud providers like Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP) are amongst the world's largest building sites. As an example, a typical Google datacenter occupies more than 185,000 m^2 (2 million square feet) of usable space – the size of 26 soccer fields[11]. Figure 27 shows a picture of the Google datacenter in Council Blufs, IA, USA. Google invested $5 billion in that datacenter site alone[12].

Figure 27: Google datacenter in Council Blufs, IA[13]

Obviously, these datacenters use an enormous amount of power. As an example, most Amazon datacenters house between 50,000 and 80,000 servers, with a power capacity of between 25 and 30 megawatts[14].

4.7.3 Location of the datacenter

Finding a good location to build a datacenter can be a nontrivial task. Many variables should be considered to determine where a datacenter could be installed.

Below is a checklist that can be used as guidance when choosing a location for a datacenter:

- Environment
    - Is enough space available to expand the datacenter in the future? The initial datacenter should be designed with enough free space and spare capacity in utilities to allow for growth.
    - Is the location vulnerable to flooding? Some countries are below sea level, are in a vulnerable delta, or are close to a river. In that case make sure the datacenter is not located at the ground floor or (worse) the basement, but for instance on the third floor.

In 2015, outside of the Amsterdam AMC hospital a large water supply pipe broke. The water flooded not only the ground floor of the hospital, but also the basement, that hosted steam systems needed to sterilize the hospital's tools. All patients in the hospital were evacuated immediately and the hospital was closed for two weeks, leading to multi-million-dollar damages.

Later, the hospital management acknowledged that putting critical systems in the basement was a design flaw in the building's architecture.

    - Is the datacenter located in a hurricane prone area?
    - What is the chance of an earthquake?
    - What is the climate like? Datacenter cooling can be easier accomplished and is much cheaper in places with a low ambient temperature with low temperature fluctuations.
    - Is the datacenter close to possible external hazards like fireworks storage, a waste dump, or a chemical plant?

- What is the crime rate? Are there many burglaries in the neighborhood? What about vandalism or the possibility of terrorism?
- Is the datacenter near an airport (chance of crashing airplanes)?
- Is the datacenter near an area that is likely to be closed because of unforeseen circumstances (like a car crash on a nearby highway, a forest fire, a military location, or a nuclear plant)?
- Is the location close to the home or office of maintenance staff, systems managers, and external expertise?
- Can the datacenter be reached easily in case of emergencies?
- Are hospitals, police, and fire fighters located in the vicinity?

- Visibility
  - Is the location of the datacenter included in public maps (like http://www.datacentermap.com)?
  - Does the building have windows? Windows are not preferred as they are easy to break into the building.
  - Are markings on the building showing that this building contains a datacenter?
- Utilities
  - Is it possible to have two independent power providers and internet providers?
  - Can cabling routes to the building be determined? Is it possible to have double power and data connections leave the building from two different places?
  - Can cabling routes inside the building be determined in a flexible way? Are there multiple paths available to the patch panels, floors, and end users?
  - Is the datacenter located in a shared building? What if the building must be evacuated? What if the power must be shut down due to maintenance activities performed by another user of the building?
  - Is enough power available to supply the datacenter? How reliable is the power supply?

- Is cheap power available? Can the datacenter use renewable energy like wind or water generated power?
- What is the available bandwidth of the external data connections? Is the datacenter close to an internet exchange point? Are dark fiber connections possible? How reliable are the data connections?

- Foreign countries
  - Can the country be reached at all times?
  - Is the country politically stable? Are there specific laws and regulations you need to adhere to or be aware of?
  - Does the country have a high level of corruption? How reliable is the staff?
  - What is the legal status of the data and the datacenter itself?

4.7.4 Physical structure

The physical structure of a datacenter includes floors, wall, windows, doors, and water and gas pipes. These components, together with the layout of the rooms around the actual computer room, are discussed in this section.

4.7.4.1 Floors

In datacenters, the floor is quite important, mostly because of the weight of the installed equipment. In a typical datacenter, the floor must be able to carry 1500 to 2000 kg/m^2. For instance, one fully filled 19" computer rack weighs up to 700 kg. The footprint of a rack is about 60x100 cm, leading to a floor load of 1166 kg/m^2. By comparison, in office buildings typically the floor can carry approximately 500 kg/m^2.

Many datacenters have raised floors. Raised floors consist of a metal framework carrying removable floor tiles. These tiles are usually 60×60 cm in size. Tiles can be lifted individually to reach cables installed under the raised floor. To lift the tiles, a "floor puller" or "tile lifter" is used, as shown below.

Picture 5: Removable tiles in a raised floor

Raised floors are typically installed at heights between 40 cm and 120 cm. Vents in the raised floor provide cool air flow to the racks placed on the floor. Under the raised floor, data and power cables are installed (usually in cable trays).

It is important to keep data cables and power cables separated from each other, as electrical current flowing through the power cables can interfere with data being sent through the data cables. A rule of thumb is to keep one phase electricity and data 20 cm apart from data cables, and 3 phase power and data cables 60 cm apart.

Not all datacenters use raised floors anymore, since raised floors have the following disadvantages:

- Raised floors are expensive.
- The total available height in the datacenter is decreased, which could lead to regulation problems and problems installing large equipment.
- The maximum floor load is limited.
- Doors and equipment loading slopes are hard to install due to the difference in floor height.
- Under the raised floor, fire, such as from a short circuit, could easily spread throughout the datacenter.

Instead of installing cables under raised floors, overhead cable trays can be used.

In either situation, cable trays can be installed with several layers. For instance, the bottom layer can be used for data copper UTP cables, the middle layer for fiber cables, and the top layer for power cables.

4.7.4.2 Walls, windows, and doors

Because of fire safety and physical intrusion prevention, walls should reach from the floor to the building's ceiling. Walls should have an adequate fire rating to serve as a physical firewall.

Windows in the outside of the building, facing the computer room, are not desirable in a datacenter. If they are present however, they must be translucent and shatterproof, and it must be impossible to open them.

Doors in the datacenter must resist forced entry and have a fire rating equal to the walls. Emergency exits must be clearly marked, monitored, and alarmed. Doors should be large enough to have equipment brought in, with a minimal width of 1 m and a minimal height of 2.10 m.

4.7.4.3 Water and gas pipes

When the datacenter is part of a larger building, water or gas pipes may have been installed under the floor, in the walls, or (even worse) above the ceiling of the datacenter. At multiple occasions, I have seen leakage from water pipes in the ceiling of a datacenter that led to damage of equipment. Datacenter operators should know where the shutoff valves are to water or gas pipes in the building.

SKIPPED TEXT

4.8 Networking building blocks

4.8.1 OSI Reference Model

The architecture of almost every network is based on the Open Systems Interconnection (OSI) standard reference model. The OSI Reference Model (OSI-RM) was developed in 1984 by the International Organization for Standardization, a global federation of national standards organizations representing approximately 130 countries.

A host or node is a component on the network, like a server, a router, a switch or a firewall. The OSI-RM consists of a set of seven layers that define the different stages that data must go through to travel from one host to another over a network. Figure 40 shows these seven layers, including some examples of implementations of that layer.

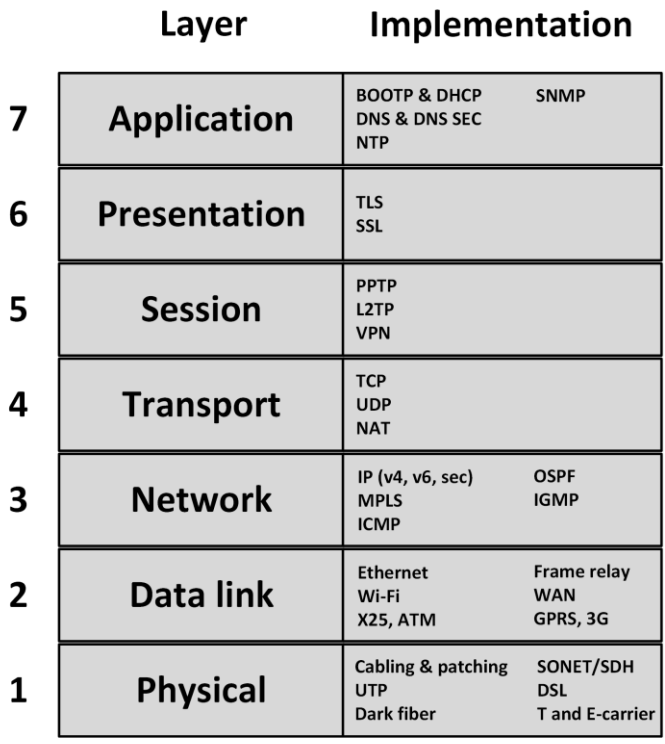


Figure 40: OSI layers

The layers can easily be recalled using the mnemonic:

***P**eople **D**o **N**eed **T**o **S**ee **P**amela **A**nderson,*

where the first letter of each word is the first letter of each layer, starting from layer one.

The main benefit of implementing the OSI stack is that it allows implementing network components independently of each other, while still ensuring all components work together. For instance, TCP/IP, which is used to send information over the internet, comprises the TCP protocol in layer 4 with the IP protocol in layer 3. Without changing the IP protocol, an UDP/IP stack can be used as well, by just changing the level 4 protocol from TCP to UDP.

Because each layer in the OSI stack can be implemented independently from the layer below and above. This provides freedom to implement the network stack

in an optimal way for a certain usage. For instance, local area networks use different building blocks than wide area networks or the internet.

Each layer's payload contains the protocol for the next layer. Consider the example in Figure 41.

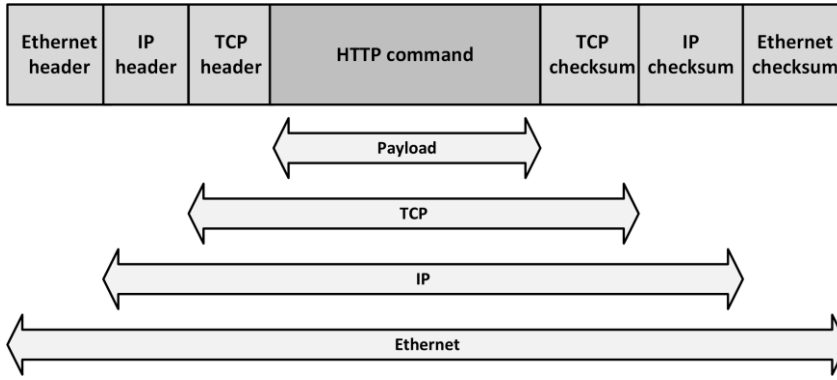


Figure 41: Frames embedded in each other

Figure 41 shows an Ethernet frame with an IP packet in it, with a TCP segment in it, with a HTTP command in it. The nesting of these protocols allows sending HTTP traffic (like web pages) to another computer using an Ethernet network in a reliable way.

This chapter is organized based on the OSI model, starting from the bottom layer and working up to the top of the stack. For each layer the most used implementations are discussed.

4.8.2 Physical layer

The physical layer defines physical hardware components of the network, such as Network Interface Controllers (NICs), copper and fiber optic cables, leased lines, cable internet, and DSL.

4.8.2.1 Cables

At the most elementary level, networking is about cables. In early networks coax cables were used to interconnect computers, but most copper-based cables today are of the twisted pair type. Apart from copper cabling, fiber optic cabling is used quite often as well.

4.8.2.1.1 Twisted pair cables

Twisted pair cables consist of paired insulated wires that are twisted around each other to prevent interference. A cable contains multiple wire pairs, that can be shielded (Shielded Twisted Pair - STP) or unshielded (Unshielded Twisted Pair - UTP). UTP is the most common cable in networking today.

Picture 11: UTP cable

Having separate pairs of wires for transmitting data (TX) and receiving data (RX) allows for full duplex communication. Full duplex communication means that data may be transmitted and received by a host at the same time.

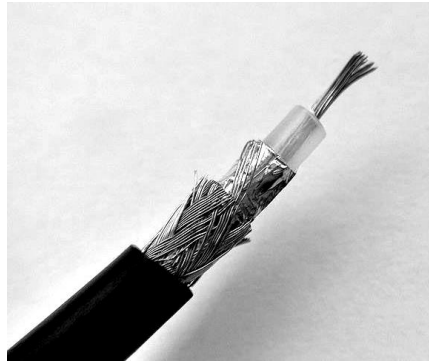
UTP comes in several quality ratings called categories. The rating is based on how tightly the copper wires are intertwined: The tighter the wind, the higher the rating and its resistance to interference and attenuation. This resistance to interference is crucial for providing higher data rates. Table 12 shows a list of today's most used categories and their maximum bandwidth.

Category	Maximum bandwidth
5 or 5e	1 Gbit/s
6	10 Gbit/s
7	10 Gbit/s
8	40 Gbit/s

Table 12: Twisted pair cables and their bandwidth

4.8.2.1.2 Coax cable

Coax cable consists of an inner conductor surrounded by a flexible, tubular insulating layer, surrounded by a tubular conducting shield.



Picture 12: Coax cable[15]

Historically, coax cable provided the highest bandwidth possible in copper cabling. It is still heavily used by cable companies, but improvements in UTP and STP cables allow higher bandwidths, eliminating coax cables for most other uses.

4.8.2.1.3 Fiber optic cable

A fiber optic cable contains multiple strands of fiber glass or plastic, that each provide an optical path for light pulses. The light source can either be a light-emitting diode (LED) or a laser.

The maximum transmission distance depends on the optical power of the transmitter, the optical wavelength utilized, the quality of the fiber optic cable and the sensitivity of the optical receiver.

Two types of fiber optic cable are most common:

- Multi-Mode Fiber (MMF)
- Single Mode Fiber (SMF)

SMF is used for long distance communication (up to 80 km), and MMF is used for distances of 500m or less, typically used in the datacenter or on a campus setup.

Light waves in Multi-Mode Fiber (MMF) are dispersed into numerous paths, also known as modes, as they travel through the cable's core – hence the name.

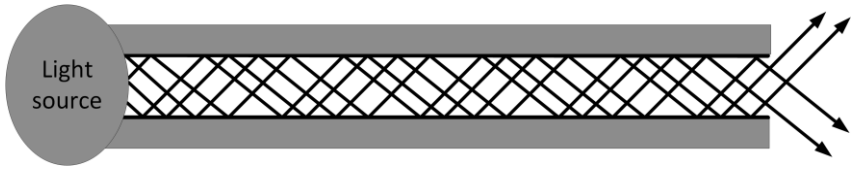


Figure 42: Multi-Mode Fiber

Single-Mode Fiber (SMF) is designed to carry only a single narrow band of ray wavelengths of light (a single mode).

Figure 43: Single-Mode Fiber

SMF requires a light source with a narrow spectral width (typically a laser). SMF has a much smaller core than MMF. The small core and single light-wave virtually eliminates any distortion that could result from overlapping light pulses, providing transmissions over long distances. SMF is more expensive than MMF, not only because of cable costs, but also because of the more expensive interface cards needed to send a single ray of light.

Using one light source, the maximum bandwidth of a fiber optics cable (both MMF and SMF) is approximately 10 Gbit/s. Using Dense Wavelength-Division Multiplexing (DWDM) the capacity of fiber optics cables can be extended. By using multiple light sources, each having a distinct color (wave length), multiple channels can travel though the fiber optics cable simultaneously. This way up to 80 channels can be created, leading to a total bandwidth of 800 Gbit/s for a single strand of fiber cable.

SKIPPED TEXT

4.9 Storage building blocks

Servers can use internal storage, but most use external storage, sometimes combined with internal storage. A model of storage building blocks is shown in Figure 70. Each building block is discussed in detail in de subsequent sections, starting at the lowest building blocks.

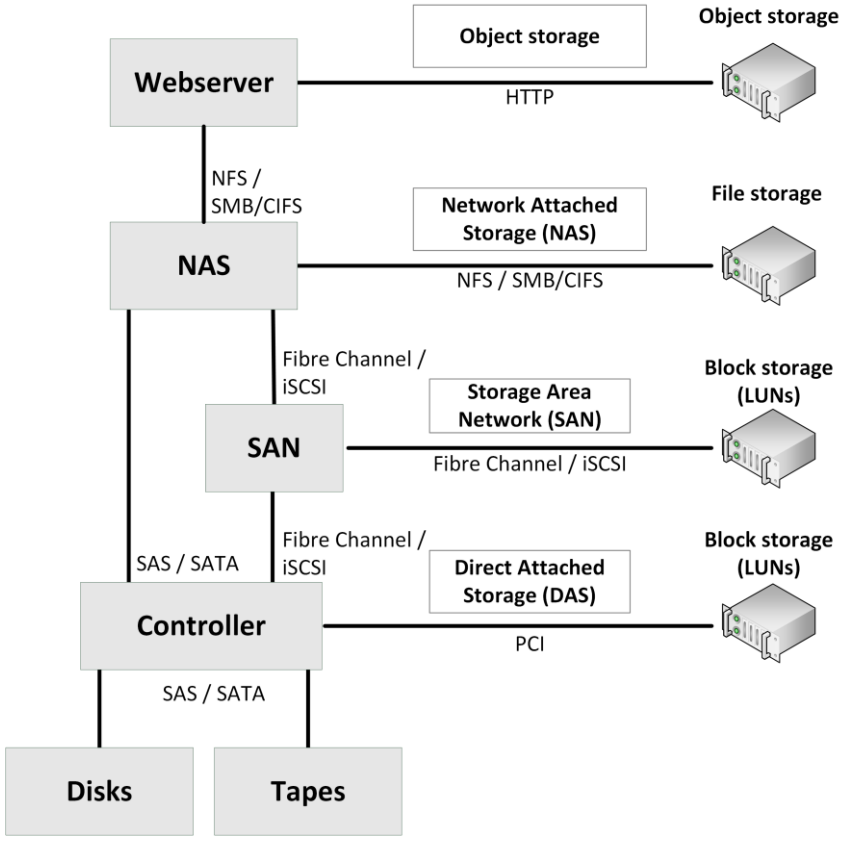


Figure 70: Storage model

4.9.1 Disks

Two types of disks are in use today:

- Mechanical hard disks
- SSD disks

Disks are connected to disk controllers using a command set, based on either ATA or SCSI.

4.9.1.1 Command sets

Disks communicate with disk controllers using a protocol based on either the ATA or SCSI command set.

Advanced Technology Attachment (ATA), also known as IDE, uses a relatively simple hardware and communication protocol to connect disks to computers (mostly PCs). For many years, ATA provided the most common and the least expensive disk interface.

Small Computer System Interface (SCSI) is a set of standards for physically connecting and transferring data between computers (mostly servers) and peripheral devices, like disks and tapes. The SCSI standard defines command sets for specific peripheral device types. The SCSI command set is complex - there are about 60 different SCSI commands in total.

The need for increased bandwidth and flexibility in storage systems made the original parallel SCSI and ATA standards an inefficient option. Serial interfaces replaced the parallel interfaces, but the disk commands are still the same.

4.9.1.2 Mechanical hard disks

Mechanical disks consist of vacuum sealed cases with one or more spinning magnetic disks on one spindle and a number of read/write heads that can move to reach each part of the spinning disks. Picture 19 shows a mechanical hard disk with its cover removed,

Picture 19: Hard disk internal mechanical construction

In today's systems, three mechanical (spinning) disk types are most common, depicted by their used interface:

- Serial ATA (SATA) disks
- Serial Attached SCSI (SAS) disks
- Near-Line SAS (NL-SAS) disks

SATA disks are low-end high-capacity disks. SATA disks are ideal for bulk storage applications (like archiving or backup) as they have a low cost per gigabyte. SATA disks are also often used in PCs and laptops. SATA disks use the SMART command set to control the disk. This command set is limited, but easy to implement.

SAS disks are relatively expensive, high end disks with spinning disk platters with a rotational speed of 10,000 or 15,000 rpm. This makes them very fast, but they typically have 25% of the capacity of SATA or NL-SAS disks.

SAS disks are high-end disks, because they have better error correction capabilities than SATA disks, and can move erroneous disk sectors to spare sectors automatically, making the disks very reliable. In addition, SAS uses the SCSI command set that includes error-recovery and error-reporting and more functionality than the SMART commands used by SATA disks.

NL-SAS disks have a SAS interface, but the mechanics of SATA disks. Because NL-SAS disks use the SAS protocol, they can be combined with faster SAS disks in one storage array. They are used for bulk storage applications as they can store much data, have a low cost per gigabyte and use much less energy than SAS disks, as they typically spin at just 7,200 rpm.

4.9.1.3 Solid State Drives (SSDs)

A Solid State Drive (SSD) is a disk that doesn't have moving parts and is based on flash technology. Flash technology is semiconductor-based memory that preserves its information when powered off. SSDs are connected using a standard SAS disk interface.

SSD's main advantage is performance. SSDs have no moving parts, so data can be accessed much faster than using mechanical disks (microseconds vs. milliseconds). Most storage vendors now offer all-flash arrays – storage systems using only SSD disks. For high-demanding Online Transaction Processing (OLTP) systems, these all-flash arrays are the preferred choice today, because of their high performance.

Picture 20: SSD disk[16]

SSDs consume less power, and therefore generate less heat, than mechanical disks. And since they have no moving parts, they generate no vibrations that could influence or harm other components, or shorten their lifetime.

The main disadvantage of SSDs used to be their price per gigabyte, which was significantly higher than that of mechanical drives. But since 2020, the prices of SSDs have fallen to the point where they are about the same price as mechanical drives. In the coming years, mechanical drives are expected to be used only as cheap, low-end storage for applications such as archiving.

Another disadvantage of SSD is that the used flash memory can only be rewritten a limited number of times – the disks "wear out" more rapidly than mechanical disks. To overcome this disadvantage, SSDs keep track of the number of times a sector is rewritten, and map much used sectors to spare sectors if they are about to wear out. It is important to monitor the wear level of heavily used SSDs, so they can be replaced before they break.

Some SSDs utilize RAID technology internally (RAID is discussed in section 10.2.3.1), to distribute data over the available flash chips on the SSD disk. The more RAID channels are available, and the bigger the number of flash chips, the faster the SSD disk can deliver data and the more reliable the SSD becomes.

Today's Non-Volatile Memory Express (NVMe) drives are capable of delivering hundreds of thousands of read/write operations and gigabytes of throughput per second. Because of this high speed, NVMe-based SSD disks are often connected directly through the PCIe bus, rather than through a separate disk controller. Technology is moving fast in this area, so more advanced flash storage technologies are expected in the forthcoming years.

4.9.1.4 Disk capacity - Kryder's law

Since the introduction of the first disk drives, physical disk sizes shrunk and disk capacity increased every year.

Figure 71 shows that the average disk capacity has followed a logarithmic increase in size for the last 30 years (note that the Y-axis is logarithmic instead of linear).

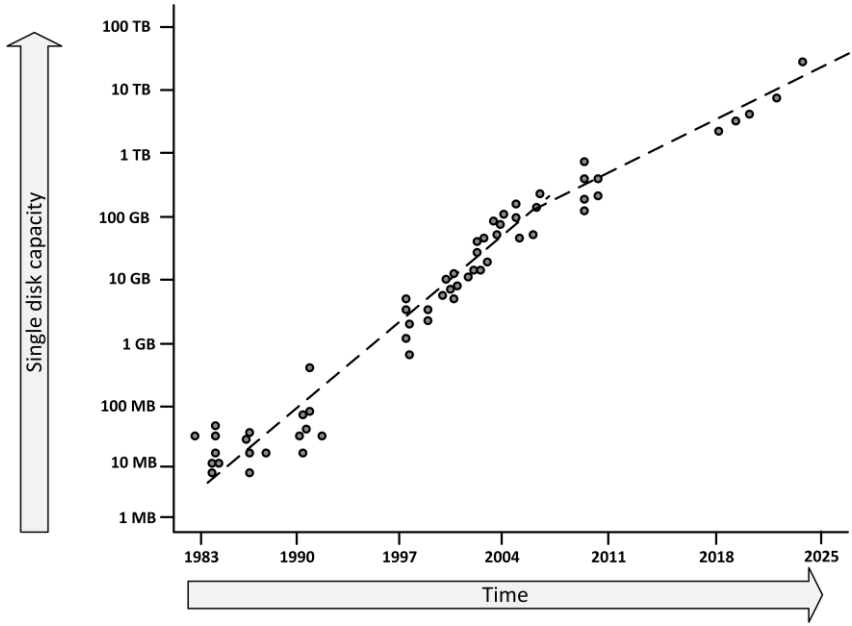
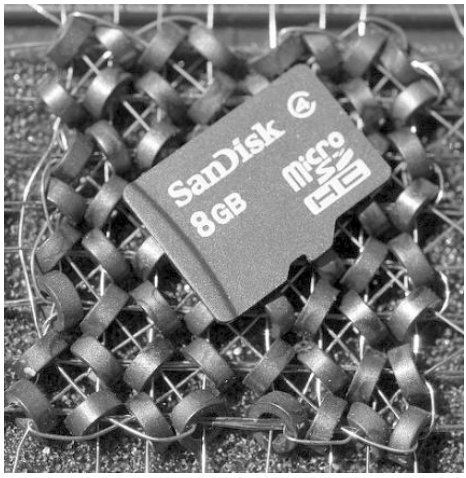


Figure 71: Kryder's law[17]

Kryder's law[18] states that "*the density of information on hard drives has been growing at a rate, increasing by a factor of 1000 in 10.5 years, which roughly corresponds to a doubling every 13 months*". Since 2005 we see a slight slowing of the curve, but it is still reasonably correct since 1983.



Picture 21: 8 bytes versus 8,000,000,000 bytes[19]

Picture 21 illustrates Kryder's law – it shows the physical size of 8 bytes of core memory from the 1960s, and a micro-SD flash card containing 8 GB of memory from the 2010's – one billion (1,000,000,000) times as much storage in 50 years.

SKIPPED TEXT

5

COMPUTE

5.1 Introduction

Compute is an umbrella term for computers located in the datacenter that are either physical machines or virtual machines. Physical computers contain power supplies, Central Processing Units (CPUs), a Basic Input/Output System (BIOS), memory, expansion ports, network connectivity, and – if needed – a keyboard, mouse, and monitor.

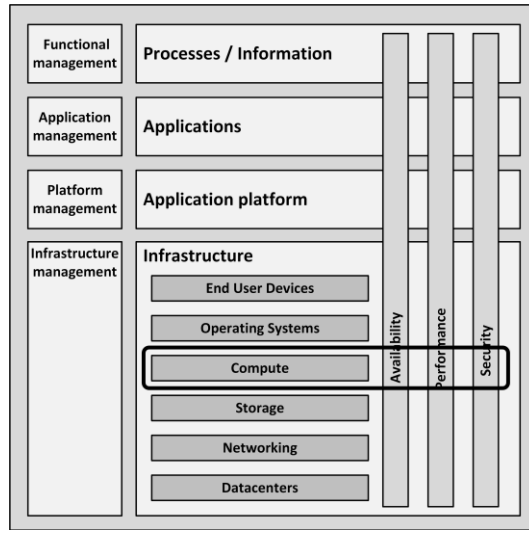
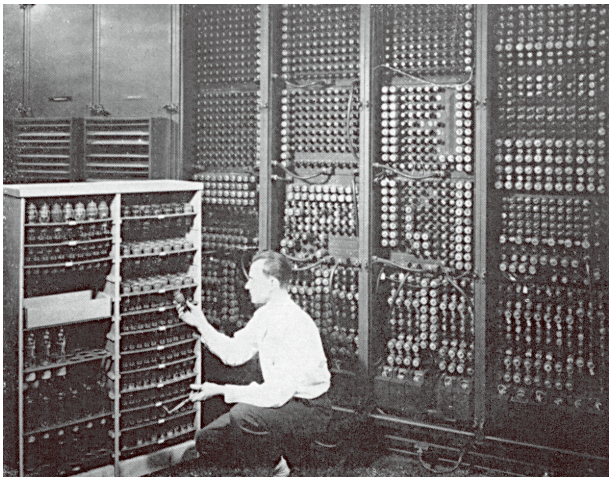


Figure 84: Compute in the infrastructure model

Originally, the word computer was used for a person who performed manual calculations (or computations). Beginning in the early 1900s, the word computer was also used for calculators. The first calculators were mechanical calculators. Computers as we know them today have two specific characteristics: they calculate, and they are programmable. Programmable computers became feasible only after the invention of punch cards, which allowed computers to process sequences of data.

The British Colossus computer, made during World War II, was the world's first programmable computer. However, its status was never publicly recognized because information about it was secret under British secrecy laws.

The first widely recognized computer was the ENIAC (Electronic Numerical Integrator And Computer). Designed in 1943, the ENIAC was funded by the U.S. Army in the middle of World War II. It was completed and fully operational in 1946 (after the war) and remained in operation until 1955. Although the original purpose of ENIAC was to calculate artillery firing tables for the U.S. Army's Ballistic Research Laboratory, the machine was first used to perform calculations for the design of the hydrogen bomb.



Picture 26: ENIAC[20]

The ENIAC could perform 5,000 operations per second, which was spectacular at the time. However, it used more than 17,000 vacuum tubes, each with a limited life span, which made the computer highly unreliable. The ENIAC got its input using an IBM punched card reader, and punched cards were used for output as well.

As a result of the invention of the transistor in 1956, in the 1960s computers started to be built using transistors instead of vacuum tubes. Transistor-based machines were smaller, faster, cheaper to produce, required less power, and were much more reliable.

The transistor-based computers were followed in the 1970s by computers based on integrated circuit (IC) technology. ICs are small chips that contain a set of transistors providing standardized building blocks like AND gates, OR gates, counters, adders, and flip-flops. By combining building blocks, CPUs and memory circuits could be created.

The subsequent creation of microprocessors decreased size and cost of computers even further, and increased their speed and reliability. In the 1980s microprocessors were cheap enough to be used in personal computers.

Today's compute systems include mainframes, midrange systems, and x86 servers. They comprise processors, memory, and interfaces, and they can be implemented as physical or virtual machines.

5.2 Compute building blocks

5.2.1 Computer housing

Originally, computers were stand-alone complete systems, called pedestal or tower computers, which were placed on the datacenter floor. Except for mainframes, most x86 servers and midrange systems are now rack mounted or placed in enclosures as blade servers.

Rack mounted x86 servers are complete machines, typically 1 to 4 Rack Units high (for more information on Rack Units, see section 8.2.8). Since they are complete machines, they need their own power cables, network cables and SAN cables.

Picture 27: A stack of rack mounted servers[21]

Blade servers, on the other hand, are servers without their own power supply or expansion slots. They are placed in blade enclosures, enabling a high server density in a small form factor. Blade servers are connected to shared power supplies, by a wiring system called a backplane.

In general, systems based on blade servers are less expensive than rack mounted servers or pedestal servers because they use the enclosure's shared components like power supplies and fans.

Picture 28: Blade enclosure with one blade partially removed[22]

A blade enclosure typically hosts from 8 to 16 blade servers and provides:

- **Shared redundant power supplies** for all blades.
- **A shared backplane** to connect all blades.
- **Redundant network switches** to connect the blades' Ethernet interfaces providing redundant Ethernet connections to other systems.
- **Redundant SAN switches** to connect the HBA interfaces on the blade servers providing dual redundant Fibre channel connections to other systems.
- **A management module** to manage the enclosure and the blades in it.

The amount of wiring in a blade server setup is substantially reduced when compared to traditional server racks, leading to less possible points of failure and lower initial deployment costs.

A set of blade servers in an enclosure typically uses less electrical power than individual rack mounted servers due to the lower overhead of the shared components in the enclosure. From a deployment perspective, blade servers are also less expensive to install, primarily because the enclosure is a wire-once component and additional blades can be added with a minimum of time and cost.

One often mentioned benefit of using blade servers is that after some years of operation, the blades can be replaced by newer and faster blades. In practice, this is not always the case.

Typically, a blade enclosure is only guaranteed to run one or two generations of server blades. Newer server blades often don't fit, or have additional power, cooling or bandwidth requirements that do not allow them to be used in an existing enclosure.

For example, a blade enclosure's power supply and backplane are designed to provide a maximum number of watts to a blade. If newer blades need more power, then they cannot be used in that blade enclosure, unless the power supplies are replaced as well (if possible).

Newer blades typically also allow for higher network and SAN throughput. The blade enclosure might not allow this, or lowers the network bandwidth to allow running newer and older blade servers together in one blade enclosure.

Enclosures are often not only used for blade servers, but also for storage components like disks, controllers, and SAN switches.

5.2.2 Processors

In a computer, the Central Processing Unit (CPU) – or processor – executes a set of instructions. A CPU is the electronic circuitry that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions[23]. Today's processors contain billions of transistors and are extremely powerful.

Picture 29: Intel Xeon Processor[24]

A CPU can perform a fixed number of instructions, such as ADD, SHIFT BITS, MOVE DATA, and JUMP TO CODE LOCATION, called the instruction set.

Each instruction is represented as a binary code that the instruction decoder of the CPU is designed to recognize and execute. A program created using CPU instructions is referred to as machine code. Each instruction is associated with an English like mnemonic to make it easier for people to remember them. This set of mnemonics is called the assembly language, which is specific for a particular CPU architecture.

There is a one-to-one correspondence of assembly language instructions to machine code instructions. For example, the binary code for the ADD WITH CARRY machine code instruction may be 10011101 and the corresponding mnemonic could be ADC.

A programmer writing machine code would write the code using mnemonics for each instruction. Then, the mnemonics are passed through a program called an assembler that performs the one-to-one translation of the mnemonics to the machine instruction codes. The machine instruction codes generated by the assembler can run directly on the CPU.

The assembler programming language is the lowest level programming language for computers and very hard for humans to create, understand, and maintain. Higher level programming languages, such as C#, Java, or Python are much more human friendly. Programs written in these languages are translated

to assembly code before they can run on a specific CPU. This process is called compiling and is done by a high-level language compiler. It allows higher level languages to be CPU architecture independent.

Actually, machine code is not the lowest level programming language, because most processors also run microcode. Microcode is a tiny program stored on the processor chip for each machine instruction. The processor executes multiple microcode instructions to implement each machine code instruction. Microcode instructions are simple instructions that generate hardware control signals.

The big advantages are that it simplifies CPU design (replacing hardware with software), it's easier to debug, and (in modern systems) you can fix many hardware bugs in the field with microcode patches.[25].

A CPU needs a high frequency clock to operate, generating so-called clock ticks or clock cycles. Each machine code instruction takes one or more clock ticks to execute. The speed at which the CPU operates is defined in GHz (billions of clock ticks per second). Because of these high clock speeds CPUs are able to execute instructions very fast. An ADD (mnemonic for addition) instruction, for example, typically costs 1 tick to compute. This means a single core of a 2.4 GHz CPU can perform 2.4 billion additions in 1 second!

Each CPU is designed to handle data in chunks, called words, with a specific size. The word size is reflected in many aspects of a CPU's structure and operation; the majority of the internal memory registers in the processor are the size of one word and the largest piece of data that can be transferred to and from the working memory in a single operation is also a word. By using large word sizes larger chunks of data can be read and written to memory in one clock tick.

While the first CPUs had a word size of 4 bits, 8-bit CPUs quickly became much more popular, where numerical values between 0 and 255 could be stored in a single internal memory register.

The first single chip 16-bit microprocessor was the Texas Instruments TMS 9900, but the 16-bit Intel 8086 quickly became more popular. It was the first member of the large x86 microprocessor family, which powers most computers today.

Today's 64-bit CPUs have registers that can hold a single value which can have 2^{64} different values. For example, an integer number between 0 and 2^{64} represents a virtual memory address. Therefore, a 64-bit CPU can address 17,179,869,184 TB of memory, as opposed to 32-bit CPUs, which can address 4 GB memory.

5.2.2.1 Intel x86 processors

Following the huge success of IBM's PC architecture in 1981, Intel CPUs became the de-facto standard for many computer architectures. The original PC used a 4.77 MHz 16-bit 8088 CPU. The follow-up model IBM PC/AT used the more advanced 16-bit 80286.

In 1985, Intel produced the 32-bit 80386 and later the 80486 processors. Since these names all ended with the number 86, the generic architecture was referred to as x86. Later, Intel processors got more marketed names like Pentium (mainly because Intel could not get the numbers patented as a name), but the architecture was still based on the original x86 design. This allowed for backwards compatibility of software; software written for the 8088 could still run on later CPU models without a change.

The latest Intel x86 model is the 24-core i9-13900K Processor, running on 3 GHz[26].

SKIPPED TEXT

5.3 Compute availability

High availability in servers can be reached by using hot swappable components, parity and ECC memory, and lockstepping.

5.3.1 Hot swappable components

Hot swappable components are server components like memory, CPUs, interface cards, and power supplies that can be installed, replaced, or upgraded while the server is running.

To prevent short circuits or electrical noise that could lead to malfunction of electronic components, the server must have dedicated circuitry to disconnect the hot swappable component. Alternatively, the server's system board may also have special connectors that physically disconnect power to a component while the component is removed.

The virtualization and operating systems using the server hardware must be aware that components can be swapped on the fly. For instance, the operating system must be able to recognize that memory is added while the server operates and must allow the use of this extra memory without the need for a reboot.

5.3.2 Parity and ECC memory

To detect memory failures, parity bits can be used as the simplest form of error detecting code. A parity bit is a bit that is added to a byte to ensure that the number of bits with the value ‘1’ in a byte is even or odd.

For instance, with even parity, when a byte of memory contains 1011 0110, the number of ones is five. In this case the parity bit stores a 1, making the number of bits even (six). When the memory contains 1001 0110 the number of ones is four. In the parity bit a 0 is stored, making the number of bits even again (still four).

```
DATA         PARITY
1001 0110    0
1011 0110    1
```

When for some reason one of the data bits or the parity bit itself is "flipped", it can be detected:

```
DATA         PARITY
0001 0110    0 -> ERROR: parity bit should have been 1!
```

Parity bits enable the detection of data errors but cannot correct the error, as it is unknown which bit has flipped.

In contrast, ECC memory not only detects errors, but is also able to correct them. ECC stands for "Error Correction Codes". ECC Memory chips use Hamming Code or Triple Modular Redundancy (TMR) as the method of error detection and correction. Hamming code can correct single bit errors occurring in data. Multi-bit errors in the same memory location are extremely rare and don’t pose much of a threat to memory systems. TMR memory, however, is able to repair two failing bits.

The BIOS of some computers, and operating systems such as Linux, can count the number of memory errors detected and corrected, to report failing memory modules before the problem becomes catastrophic.

Memory errors are proportional to the amount of RAM in a computer as well as the duration of operation. Since servers typically contain many GBs of RAM and are in operation 24 hours a day, the likelihood of memory errors is relatively high and hence they require ECC memory.

5.3.3 Virtualization availability

All virtualization products provide failover clustering. Since the virtualization layer has no knowledge of the applications running on the virtual machine's operating system, failover clustering on the virtualization level can only protect against two situations:

- A physical hardware failure.
- An operating system crash in a virtual machine.

When a physical machine fails, the virtual machines running on that physical machine can be configured to restart automatically on other physical machines. And when a virtual machine crashes, it can be restarted automatically on the same physical machine.

Some virtualization products provide monitoring of the operating systems from within the virtual machines' operating system. For instance, VMware provides the VMware-tools application running inside the operating system of the virtual machine. It monitors, among other things, if the operating system is still working. When the operating system crashes, the VMware tools are not reachable anymore and VMware will restart the virtual machine automatically.

Since failover clustering on the virtualization layer cannot protect against application failures (like a crashed application process or service), these should be handled by the operating system layer. See the chapter 12 on operating systems for more details.

Both VMware (vSphere with HA/FT) and Citrix (XenServer with Marathon everRun) also provide lockstep technology to keep two virtual machines in sync, effectively providing redundant operating systems. This technology, however, has some technical limitations and uses quite a bit of network bandwidth.

5.3.3.1 Admission

To cope with the effects of a failure of a physical machine, a spare physical machine is needed. For this setup to work, all hypervisors are placed in a virtualization cluster, so they are aware of each other. The hypervisors on the physical machines check the availability of the other hypervisors in the cluster.

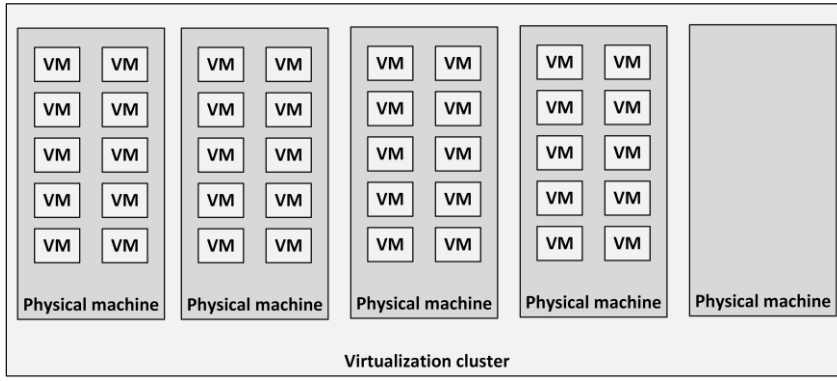


Figure 101: Using a spare physical machine

In Figure 101, one physical machine is running as a spare to take over the load of any failing physical machine. Under normal conditions the spare server is not doing any work.

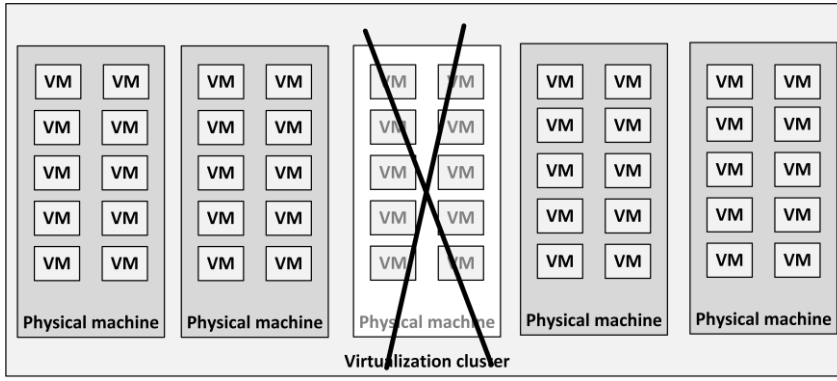


Figure 102: Failing physical machine

When one physical machine fails (Figure 102), the virtual machines running on it are automatically restarted on the spare physical machine.

An alternative is to have all physical machines running at lower capacity. For instance, when 5 machines are in a virtualization cluster, and each machine could host ten virtual machines, the total load of all servers should be $4 \times 10 = 40$ virtual machines. Instead of having one spare server running, the workload can also be spread over all machines, each hosting $\frac{40}{5} = 8$ virtual machines.

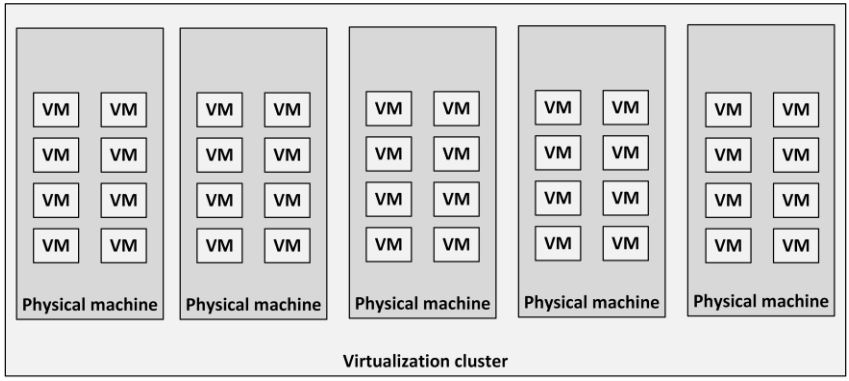


Figure 103: All machines used

This way all resources are used as much as possible since the hypervisor will provide extra resources like RAM and CPU to the virtual machines automatically, even though it is still possible to handle a failure of a physical machine. In that case the four remaining physical machines still have the capacity to run 8 extra virtual machines and the virtual machines that ran on the failed physical machine can automatically be restarted on the other physical machines (Figure 104).

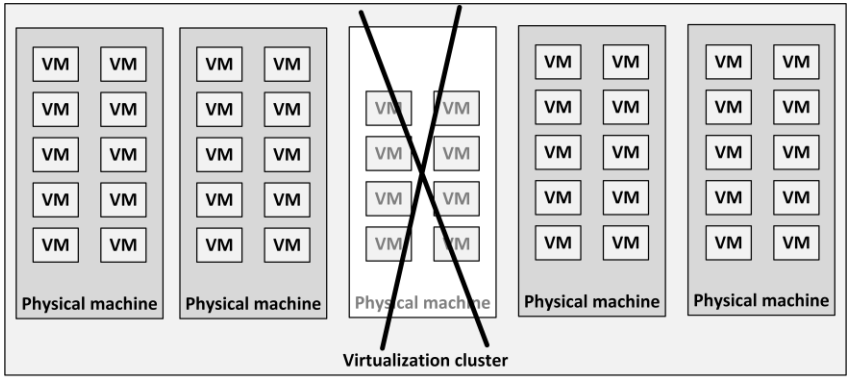


Figure 104: Failing machine when all machines were used

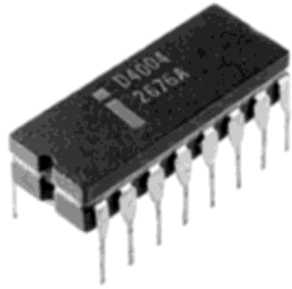
5.4 Compute performance

The performance of computers is dependent on the architecture of the server (which is described in earlier sections), the speed of the memory and CPU, and the bus speed.

5.4.1 Moore's law

Today, all computers use microprocessors as their Central Processing Unit (CPU). Before the invention of microprocessors, a single CPU was built using one or more circuit boards, containing large numbers of Integrated Circuits (ICs). Each IC contained from tens to a few hundred transistors.

In 1971, Intel released the world's first universal microprocessor, the 4004. A microprocessor is nothing more than a very complex IC, combining the functions of all the individual ICs and the circuitry needed to create a CPU, effectively creating a processor on a chip.



Picture 36: Intel 4004 microprocessor[27]

The 4004 chip itself was 3 mm wide by 4 mm long and consisted of 2,300 transistors. The chip was mounted in a DIP package with 16 connection pins (the DIP package was much larger that the chip itself of course). Coupled with one of Intel's other products, the RAM chip, the microprocessor allowed computers to be much smaller and faster than previous ones. The 4004 was capable of performing 60,000 instructions per second, which was about as much as the ENIAC computer that filled a complete room and weighed several tons.

Since the introduction of the first CPU in 1971, the power of CPUs has increased exponentially. This makes today’s computers much more powerful than we could possibly have imagined forty years ago.

Moore's law states that the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years. This trend has continued for more than half a century now. The law is named after Intel's co-founder Gordon E. Moore, who described the trend in his 1965 paper "Cramming more components onto integrated circuits"[28], when he worked at Fairchild.

Over the years, the number of transistors on a CPU raised from 2,300 on the first CPU (the 4004 in 1971) to 100,000,000,000 (100 billion) on an Intel Alder Lake hybrid processor in 2022. This is an 43 million-fold increase!

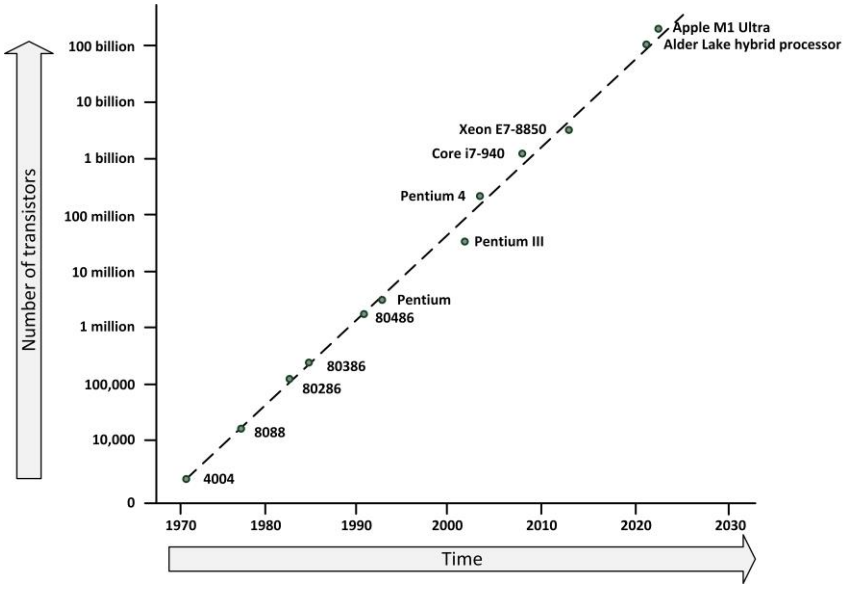


Figure 105: Moore's law

Figure 105 clearly shows the trend. Please note that the vertical scale is logarithmic instead of linear, showing a 10-fold increase of the number of transistors in each step.

Note that Moore's law only speaks of the number of transistors; not the performance of the CPU. The performance of a CPU is dependent on a number of variables, like the clock speed, the use of caches and pipelines, and the width of the data bus. When we look at the performance gain, we see a doubling of CPU performance every 18 months; even faster than what Moore's law states.

Obviously, Moore's law cannot continue forever, as there are physical limits to the number of transistors a single chip can hold. Today, the connections used

inside a high-end CPU have a physical width of 5 nm (nanometer). This is extremely small – the size of 24 atoms (the diameter of an atom is of the order of 0.21 nm[29])!

When designing an infrastructure, it sometimes makes sense to take Moore's law into account by not purchasing too much spare capacity in advance. By purchasing and implementing new servers "just in time", the purchased server will have twice the processing capacity of a server you could have purchased 18 months earlier, for the same price. Therefore, to get the full benefits of Moore's law, the infrastructure (management) must be designed to handle just in time upgrades.

SKIPPED TEXT

5.5 Popular operating systems

5.5.1 z/OS

One of the first operating systems was IBM's OS/360, introduced in 1964. It was a batch processing system, created for the IBM system/360 mainframe computer. Later, OS/360 MFT (Multitasking with a Fixed number of Tasks) and OS/360 MVT (Multitasking with a Variable number of Tasks) provided multitasking to mainframes. The successor of OS/360 was OS/370, which introduced the concept of virtual memory in 1972 (see section 11.2.5.4 for more information on virtual memory).

MVS, released in 1974, was the primary operating system on the System/370 and System/390. The 64-bit version of MVS for the zSeries mainframes was named z/OS and was introduced in 2000. IBM’s z/OS is now the most used mainframe operating system. It runs on IBM mainframes only.

Extreme backward compatibility is one of z/OS's main design philosophies: programs written for MVS in 1974 can still run on today's z/OS without modification.

Reading and writing a tremendous amount of data and performing relatively simple calculations on it (for example, "read in these 400,000 records of data, do 6 calculations on each, and then output 400,000 separate reports") is a typical use of mainframes running z/OS.

While z/OS is still most used for this type of batch processing, it can be used interactively as well. A system running z/OS can support thousands of interactive users simultaneously.

z/OS doesn't always have the default settings that we take for granted on other systems. Most of the settings are to be set by systems managers. Many settings and details are site-specific, so a new user on a particular z/OS system needs to find his way around the system first in order to be able to work with it.

5.5.2 IBM i (OS/400)

IBM i is an operating system only used on IBM's Power Systems (previously called iSeries and AS/400 systems) midrange systems.

In 1969, eight years after DEC introduced the PDP-1, IBM introduced its first minicomputer: The System/3. Because the system was relatively expensive and was less advanced than the DEC systems, the System/3 was never very popular. The IBM System/32, introduced in 1975, and its successor, the System/34, were also not very popular, but the System/38 (in 1978) and the System/36 (in 1983) were.

Users found the System/36 and its operating system easy to use. IBM kept this in mind when designing the OS/400 operating system for the new series of AS/400 midrange systems. Over the years, the name of the operating system has changed from OS/400 to i5/OS to IBM i[30].

One of the biggest advantages of IBM i is its completeness. Communications, transaction processing, and system security were implemented as intrinsic parts of the operating system from the start. IBM i also has a relational database manager built in as an integral part of the operating system. Features for the implementation and maintenance of data security are implemented natively as part of the operating system.

The latest version is known officially as *IBM i 7.5*[31].

5.5.3 UNIX

UNIX is a multitasking, multi-user operating system, originally created by AT&T. In 1969, at Bell Labs, Ken Thompson, Dennis Ritchie, and others got hold of a little-used PDP-7 system. They used the machine to create a new time-sharing multi-user multitasking operating system, based on earlier work on a system called MULTICS.

The first UNIX version was written entirely in PDP assembler, which made it highly dependent on the hardware. In 1973, UNIX was rewritten in the new C programming language (C was also created by Dennis Ritchie, together with

Brian Kernighan, which makes UNIX and C are very much related to each other). This made UNIX portable to multiple types of computer hardware.

In 1975, version 6 was the first to be widely available outside of Bell Labs (later AT&T). In 1982, UNIX was licensed to a number of computer manufacturers, including Sun Microsystems and Hewlett-Packard. Most of these vendors started to market their own UNIX versions based on the original UNIX source code. They adapted the code to meet their own hardware and software requirements.

In early 1993, AT&T sold its UNIX System Laboratories to Novell. In 1994 Novell transferred the rights to the UNIX trademark and the specification to The Open Group. Subsequently, it sold the source code and the product implementation (called UNIXWARE) to SCO.

Because UNIX is written almost entirely in the C programming language, and because the source code is published, it has been ported to a wide variety of machine architectures.

Originally, AT&T registered "UNIX" as a trademark, so although anyone could create their own version of UNIX and market it, they were not allowed to call it UNIX. As a result, vendors came up with different names for their UNIX flavors:

Vendor	**UNIX flavor**
IBM	AIX
Oracle/Sun	Solaris
HP	HP-UX
Apple	Mac OS X (built on FreeBSD, discussed in the next section)

Table 21: UNIX flavors

These versions are 90% the same, but have some minor differences, like the wording of error messages, the order of commands used to start up the machine, or the location of certain files.

Each of these flavors needs specific hardware. HP-UX only runs on HP Integrity systems, and these systems cannot run for example AIX.

Applications running on a particular flavor of UNIX cannot run on another flavor without (at least) recompiling. This means that software vendors must provide separate versions of their applications for each flavor of UNIX.

UNIX popularized the hierarchical file system with nested subdirectories, a feature now implemented in most other operating systems as well. All files and directories appear under the so-called root directory "/", even if they are stored

on different physical disks. UNIX has no concept of drive letters; drives are mounted on a branch in the directory tree, providing disk space for that particular branch.

The UNIX philosophy is to use a large set of small tools that do only one thing, and do it very well. To perform complicated tasks, commands can be combined using a system called pipes. Pipes feed the output of one command to the input of another command, without storing the intermediate result. For instance, the UNIX command:

```
ls | sort
```

prints a sorted list of files on the screen. The pipe sign "|" ensures that the output of the "ls" command is routed (as input) to the "sort" command. Since after the sort command there is no further pipe specified the final output is send to the standard output system: the screen.

Of course, this is a very simple example. In practice these chains of piped commands can get very long and complex.

In UNIX, everything is treated as a file, even printers, modems, the keyboard and the screen. This allows piped commands, for instance, to use typed input from the keyboard, process them using some application, and have the output send automatically to a printer.

5.5.4 Linux

Linux is a UNIX-like operating system, but is not derived from the UNIX source code. Instead, it was developed independently by a group of developers in an informal alliance on the internet as a free operating system for the x86 platform.

In 1987, Andrew Tanenbaum, who was a professor of computer science at the Vrije Universiteit, Amsterdam in the Netherlands, wrote a clone of UNIX, called MINIX, for the IBM PC. He wrote MINIX especially for his students to teach them how an operating system worked. Tanenbaum wrote a book[32] that not only listed the 12,000 lines of MINIX source code, but also described each important part of the source code in detail, including the theory about why it was programmed the way it was.

Linus Torvalds, at the time a student at the University of Helsinki, studied MINIX in an operating system course and bought a PC to try it. In 1991, Torvalds wanted to explore the multitasking possibilities of the new Intel 80386 CPU in his PC and decided to create a small multitasking, multi-user operating system himself with the help of the internet community. On USENET, he asked developers on the internet to help him with the development[33]. Because of the open source nature of Linux many developers contributed with kernel patches, device drivers, and additions like multilingual keyboards, floppy disk drivers, support for video card devices, and much more.

It is important to understand that Linux is actually only an operating system *kernel*. Today's Linux distributions consist of the Linux kernel and its drivers, and the GNU project's applications, libraries, compilers, and tools.

The GNU project (GNU is a recursive acronym for "GNU's Not UNIX!") was launched in 1984 by Richard Stallman, to develop a free UNIX-like operating system. By 1990, the GNU project had recreated all the major components of the UNIX-like system except one – the kernel. Combining Linux with the almost-complete GNU system resulted in a complete operating system: the GNU/Linux system.

Linux and the GNU tools are licensed under the GNU General Public License, ensuring that the all source code will be free for all to copy, study, and to change.

Soon, commercial vendors showed interest. Linux itself was, and still is, free. What the vendors did was compiling the source code, adding some tools and configurations of their own, and releasing it in a distributable format. Red Hat, SuSe, Ubuntu and Debian are some of the best-known Linux distributions. Extended with Graphical User Interfaces (like KDE or GNOME), user-friendly Linux distributions became very popular.

Today Linux is a very mature operating system. Companies like Red Hat and SUSE sell professional Linux distributions including support contracts.

Linux is used everywhere – in servers, workstations, mobile devices, all Android smartphones, and appliances like set-top boxes, firewalls and NAS devices. Almost all of the internet services run on Linux. Ninety-five per cent of the supercomputers listed in the top 500 list of the fastest computers in the world[34] are running Linux.

While Linux typically runs on x86 servers or ARM based devices, some Linux distributions can be used on IBM mainframes, running in virtual machines.

Since Linux's design is derived from UNIX's design, Linux commands and scripts are to a large degree similar to those of UNIX. Linux not only uses the same (well-known) commands, but also the same file structure, scripting language, pipes, etc. This allows experienced UNIX systems managers to use Linux without the need for much extra knowledge. Porting systems from UNIX to Linux is therefore generally much easier than porting them to for instance Windows.

5.5.4.1 Linux support

Linux is created as an open source project. This means that the source code of Linux is published and freely available. While this allows users to change the source code to their needs, this is hardly ever done, due to the complexity of the Linux source code and the limited benefits of changing the code.

Most organizations demand professional support for their software. And although Linux can be downloaded from the internet for free, professional support is certainly not free. Most Linux distribution vendors, like Red Hat and SUSE, and some independent vendors, offer support contracts for Linux.

SKIPPED TEXT

6

END USER DEVICES

6.1 Introduction

Humans interact with applications using end user devices. Typical end user devices are desktop PCs, laptops, virtual desktops, mobile devices like phones and tablets, and printers.

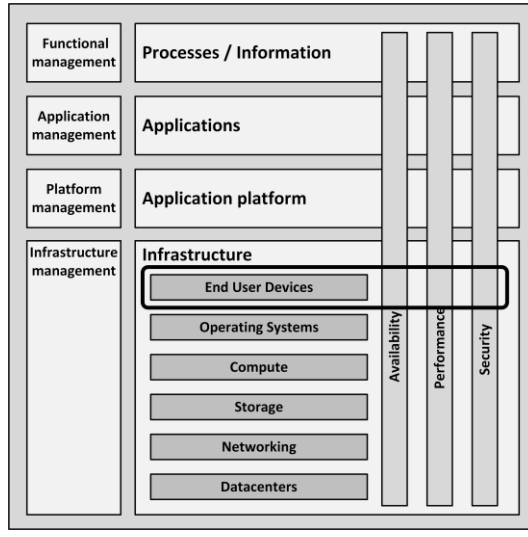


Figure 120: End user devices in the infrastructure model

The first end user devices were teletypes. Teletypes were electromechanical typewriters that provided a user interface to early computers, sending typed data to the computer and printing the response.

Picture 37: Teletype[35]

Later, electronic terminals replaced the teletypes. Terminals provided a monitor screen instead of printed paper, allowing full screen editing and instant output. Terminals were "dumb", as they did not have their own processing power. They relayed typed-in commands to the mainframe or midrange computer and the computer sent data back to the terminal to be displayed. Terminals were used for decades to interact with mainframe and midrange computers.

In 1981, IBM introduced the Personal Computer (PC). The IBM PC became the de facto end user device in many office environments, allowing office workers to have full control over their own computer for the first time.

Picture 38: The original IBM PC-XT[36]

IBM developed the PC in about a year. To achieve this, they decided to build the machine with "off-the-shelf" parts from a variety of manufacturers. They also decided on an open architecture, enabling other manufacturers to produce and sell peripheral components and compatible software without having to purchase licenses. IBM even sold an IBM PC Technical Reference Manual which included complete circuit diagrams and a listing of the ROM BIOS source code.

The result was that many parties copied the PC – the so-called PC clones. These clones (or IBM-compatible PCs) used the same architecture, used the same chipset as the IBM PC, and used reversed-engineered BIOS software (because even though the source code was published, it was still copyrighted). This allowed clones to run unmodified IBM software. One of the first and most successful companies building clones was Compaq, which would later become part of HPE.

All of the IBM PC software was developed by third parties. The most influential one being Microsoft that provided the DOS operating system and office tools like Word and Excel.

IBM was a major computer manufacturer long before the introduction of the PC. Apple, on the other hand, was founded by two hobbyists. In 1984, Apple introduced the Apple Macintosh. It was the first commercially successful personal computer to feature a mouse and a GUI rather than a command line interface. It was designed to be used by consumers, and not as an office tool.

Both the Mac and the PC evolved over time to become much faster. Color video screens and sound boards became the norm, and laptops became the most used form factor.

6.2 End user device building blocks

End user devices can be categorized as:

- Desktop PCs
- Laptops
- Virtual desktops
- Mobile devices
- Printers

All of these categories are discussed in the following sections.

6.2.1 Desktop PCs and laptops

The most used end user devices today are desktop and laptop computers based on Intel's x86 architecture, mostly referred to as PCs. While Apple iMacs also run on the x86 platform, according to Statcounter[37], Microsoft Windows is the most used operating system for desktop and laptops at 76%, followed by Apple's macOS at 16%, and Linux-based operating systems at 5%.

Over the years, PCs have become very powerful. This enables them to run complex software and to store relatively large amounts of data. But because of the sheer complexity of the PC itself, the very advanced operating systems, the amount of locally installed software, and the performance, availability, and security issues related to all of these aspects, many organizations are searching for more cost-effective and simple solutions.

But people are attached to their PCs. The term personal computer is still correct – most users feel their PC is their personal tool that systems managers should not tamper with. This is one of the main reasons why the adoption of alternatives like thin clients (see 13.3.4) has never been as successful as it could have been.

Nowadays, most laptops are as powerful as desktop PCs. And because users can take them home or use them on the road, they are even more "personal" than desktops. Laptops, however, have some disadvantages compared to desktop PCs, like:

- Laptops frequently get lost or stolen. On average, 10% of the laptops are lost or stolen during their life cycle[38]. These laptops must be replaced, the user cannot work in the meantime, and data on the laptop that was not backed-up is lost.
- Laptops break more easily than desktops, because they are more vulnerable to drops, bumps, coffee spills, etc.
- Since most laptops are taken home every night, the chance of illegal or malicious software being installed on the laptop is much higher than on a desktop PC in the office.

When used in the office, laptops are often connected to a docking station (also known as a port replicator) using a USB-C cable. The docking station provides a number of external ports for connecting a keyboard, mouse, camera, speakers, and microphone, as well as one or more displays. The USB-C cable can also charge the laptop's battery when the laptop is connected to the docking station.

6.2.2 Mobile devices

Mobile devices in the context of this book are devices that connect to the IT infrastructure using wireless public or public Wi-Fi networks. Typical mobile devices are smartphones, tablets, and smart watches.

While the computing power of some mobile devices is getting comparable to desktop and laptop computers, mobile devices have some specific properties that infrastructure architects must be aware of.

Mobile devices typically connect to the IT infrastructure using public networks based on for example LTE technology (as explained in 9.3.3.6). The bandwidth of these connections is lower than that of Wi-Fi and wired Ethernet connections. Also, connection speed can heavily fluctuate as the users move around, and it sometimes can fluctuate quite fast when the mobile device is used inside a car or train. The reliability of the connections is therefore worse than that of Wi-Fi or wired Ethernet connections. When moving around, connections sometimes drop for short periods of time or drop altogether. Signal noise can force resending large numbers of network packets. Apps running on mobile devices are specially designed to handle these characteristics.

Another limitation of mobile devices is the small form factor forcing limited keyboard and screen sizes. Applications' user interfaces must be re-engineered to handle these smaller sizes.

6.2.3 Bring Your Own Device (BYOD)

In many cases, organizations use standard PCs or laptops with a limited set of business software. In contrast, users at home have access to fast, sexy laptops of the brand they like, tablets and smart phones that allow them to run thousands of highly attractive apps and they have fast broadband internet connections at home that are often faster than the shared network in the office.

To attract new employers and because people will take their personal device to the office anyway, most organizations are now confronted with a concept called Bring Your Own Device (BYOD).

BYOD allows people to bring personally owned – typically mobile – devices to the office, to use them to access the organization's applications and data, as well as their personal applications and data.

The BYOD concept creates a conflict of interests. To optimize stability of the organization's infrastructure and security, systems managers need to fully control the end user device, while the owners of the devices want full freedom. And since the user paid for the device (they brought their *own* device), it will not be acceptable for users to have systems managers erase the device (including all family photos) in case of an incident, or to have personal data visible to the systems managers.

Virtualization techniques can be used to create isolated environments on these devices. Some solutions implement a hypervisor on the device that runs two virtual machines:

- One virtual machine that has access to the organization's data and applications and is fully managed by the organization's systems managers. This virtual machine is managed using Mobile Device Management (MDM) software that can monitor, maintain and secure the virtual machine. If necessary, the organization's managed virtual machine can be wiped remotely to remove all sensitive data.
- One virtual machine that is owned and managed by the end user. This machine runs whatever applications the user wants (browsers, social network clients, games, streaming music players, video players, etc.).

Both virtual machines use the same underlying hardware like network connectivity, touch screen, GPS, compass, and the sound system. But since both virtual machines are run on top of a hypervisor, no sensitive data will be available from the user's managed virtual machine.

SKIPPED TEXT

6.2.4 End user authorizations and awareness

End users should not be able to remove important software or alter system files or log files on their devices. Therefore, they should not have (access to) the administrator password of their device. When users need to install software (which is a frequent requirement in practice, especially in developer environments), they could be given the right to do so, without giving them the administrator password of their device

BIOS passwords should be used on laptops and desktops to further increase security. BIOS settings should be applied to prevent booting from USB memory devices.

But the security issue with end user devices is not so much a matter of the device as it is a matter of the end user. Users need to be aware of common security guidelines including the possibility of social engineering, using strong passwords and knowing how to handle sensitive data.

PART IV
–
INFRASTRUCTURE MANAGEMENT

We live in a society exquisitely dependent on science and technology, in which hardly anyone knows anything about science and technology.

Carl Sagan, American astronomer, 1990

7

INFRASTRUCTURE DEPLOYMENT OPTIONS

7.1 Introduction

Where the chapters in Part III were about technological infrastructure building blocks, this part IV is about the systems management processes. It explains the various ways infrastructure can be deployed, the steps to deploy an infrastructure, how automation can replace manual configuration, and how to manage the infrastructure and deploy applications. Finally, it describes the steps to decommission an infrastructure at the end of its life cycle.

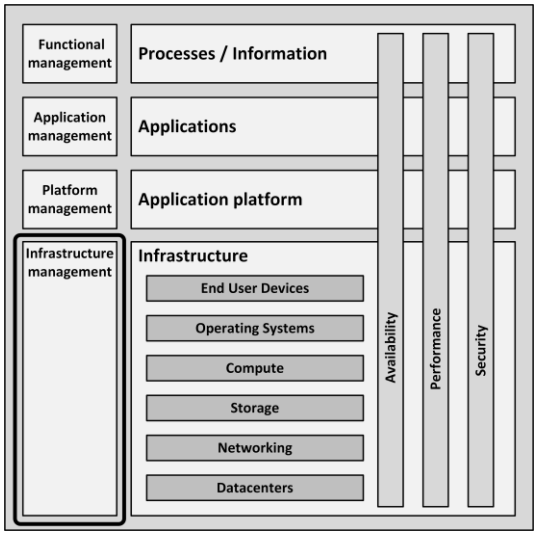


Figure 126: Infrastructure management

This chapter discusses how to select the best deployment option for an infrastructure.

7.2 Hosting options

Infrastructure can be hosted on-premises, in a colocation, deployed in a public cloud, or the full infrastructure management can be outsourced.

With **on-premises hosting**, infrastructure components run on the premises of the organization using the infrastructure. This can be in the datacenter of an existing building, or in a dedicated, specially designed datacenter building.

As the datacenter is implemented in an organization owned building, the building must have enough space, an uninterruptable power supply (UPS), options to install sufficient cooling, fire prevention and detection, external redundant network capabilities with enough bandwidth, and sufficient floor loading capacity (see section 8.2.4.1 for more details on datacenter requirements).

Two major drawbacks of on-premises hosting are:

- Typically, on-premises datacenters don't scale well, especially if they are embedded in existing (office) buildings.
- As the organization owns and runs their own datacenter, it must have enough knowledge and staff available to manage the datacenter.

In contrast, a **colocation** is a third party dedicated datacenter where racks, floor space, and network bandwidth can be rented. A colocation provides power, cooling, and physical security, and hosts and connects customer owned infrastructure components. Colocation racks are empty – all infrastructure components must be provided and managed by the organization renting the colocation racks.

Organizations can also choose to use **public cloud computing**. Depending on the deployment model chosen, the organization delegates more or less systems management. With IaaS, the organization has to do most of the management itself, while with SaaS it has to manage the least.

An organization can also decide to **outsource their entire infrastructure**. Full infrastructure outsourcing is a subcontracting service in which some third-party purchases, deploys, hosts, and manages the infrastructure, and performs its lifecycle management. The outsourcing is managed using Service Level Agreements and typically has a very rigid change management process. Outsourcing frees the organization from investing in hardware – only leaving operational cost. The outsourcing organization must have a demand organization and process in place in order to manage the outsourcing party, but it can be freed from internal infrastructure systems managers.

7.3 (Hyper) Converged Infrastructure

In a traditional infrastructure deployment, compute, storage and networking are deployed and managed independently, often based on components from multiple vendors. In a *converged* infrastructure, the compute, storage, and network components are designed, assembled, and delivered by one vendor and managed as one system, typically deployed in one or more racks.

A converged infrastructure minimizes compatibility issues between servers, storage systems and network devices while reducing costs for cabling, cooling, power and floor space. Scaling up a converged infrastructure requires the deployment of additional racks.

Where in a converged infrastructure the infrastructure is deployed as individual components in a rack, a *hyperconverged* infrastructure brings together the same components within a single server node.

A hyperconverged infrastructure (HCI) comprises a large number of identical physical servers from one vendor with direct attached storage in the server and special software that manages all servers, storage, and networks as one cluster running virtual machines. The technology is easy to expand on-demand, by adding nodes to the hyperconverged cluster.

Hyperconverged systems are an ideal candidate for deploying VDI environments (see section 13.3.3), because storage is close to compute (as it is in the same box) and the solution scales well with the rise in the number of users.

A big advantage of converged and hyperconverged infrastructures is managing only one vendor, that provides hardware, firmware, and software. Vendors of hyperconverged infrastructures make all updates for compute, storage and networking available in one service pack and deploying these patches is typically much easier than deploying upgrades in all individual components in a traditional infrastructure deployment.

Drawbacks of converged and hyperconverged infrastructures are:

- **Vendor lock-in** – the solution is only beneficial if all infrastructure is from the same vendor.
- **Scaling can only be done in fixed building blocks** – if more storage is needed, compute must also be purchased. This can have a side effect: since some software licenses are based on the number of used CPUs or CPU cores, adding storage also means adding CPUs and hence leads to extra license costs.

7.4 Private cloud

A private cloud, also known as a software-defined datacenter (SDDC), is an architecture in which all infrastructure resources – compute, storage and networking – are virtualized, and can be configured using software APIs.

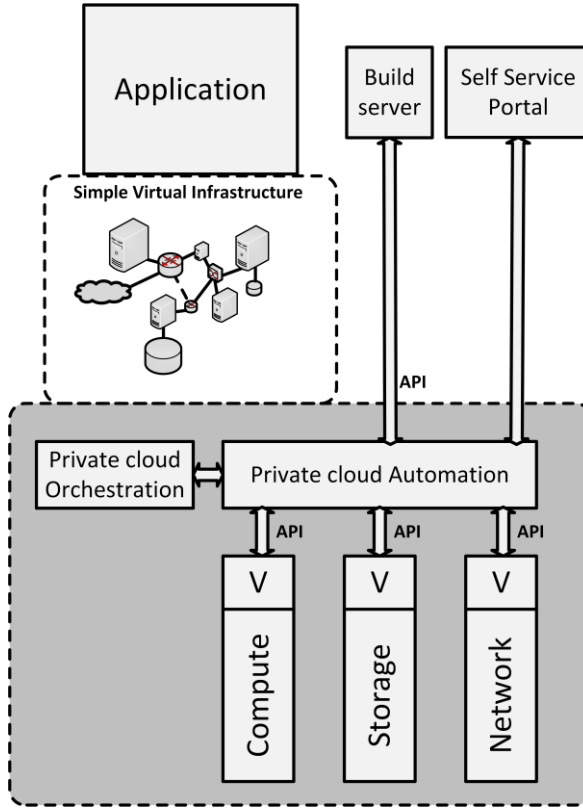


Figure 127: Private cloud

As shown in Figure 127, a private cloud is an enterprise infrastructure, where all resources are virtualized and managed by automation and orchestration software. A private cloud is not a cloud in the pure sense of the word – it has limited scaling and there is no pay per use – but its software is comparable to IaaS services of public clouds.

A private cloud is characterized by automation, orchestration, and abstraction of resources into software and code. By nature, code is more reliable than humans, which means that compared to a traditional datacenter, a private cloud is more secure and more agile. Changes are managed by an automated workflow, where an orchestrated change can lead to a number of automated changes in various resources.

A private cloud enables developers, DevOps teams and systems managers to create and deploy new infrastructures using either a manual self-service portal, or a combination of a build server and APIs. It allows the user to request the

desired infrastructure components, their sizing to meet performance demands, and their required availability; and automatically configures the private cloud components to deliver a secured infrastructure implementation. The private cloud software also provides tools for costing, logging, reporting, scaling (up and down), and decommissioning of the infrastructure resources.

Examples of private cloud automation and orchestration products are OpenStack's Horizon, IBM Cloud Orchestrator, and VMware vRealize.

A private cloud is not the solution for all problems – there are many applications that need a much more custom-designed infrastructure than the standard private cloud building blocks can deliver. Examples of these applications are SAP HANA, high performance databases, OLTP, high secure bank or stock trade transaction systems, and SCADA systems.

7.5 Public cloud

An organization may choose to build their entire infrastructure in – or migrate to – a public cloud provider. Especially in the case of a green field situation, such as a startup company, hosting the entire virtual infrastructure in the public cloud could be a viable option. Usually one of the major cloud providers is chosen, such as Amazon's AWS, Microsoft Azure or Google's GCP.

Another good reason to move to the public cloud is the use of innovative technology. Cloud providers can innovate much faster than most other organizations due to their scale and financial buffers. Customers can easily take advantage of these innovations, which not only become available quickly, but are also immediately production-ready.

7.6 Hybrid cloud

Most organizations do not choose to migrate all of their existing infrastructure to the public cloud at once. Over the years, many organizations have built a very complex landscape of infrastructure components, applications, and connections that cannot be moved overnight. In many cases, it is not cost effective to migrate an entire datacenter as-is via a lift and shift migration, which typically results in high cloud operating costs. In addition, the organization receives little value from the cloud migration. As a result, a phased approach is often taken, with some of the infrastructure remaining on-premises and some migrated to the public cloud. Because the on-premises components need to communicate with the components in the public cloud, a connection must be established between the on-premises datacenter and the public cloud provider. This is called a hybrid cloud.

A hybrid cloud often remains in place for several years, because it can take a long time to completely phase out the on-premises environment for a variety of reasons.

Some drawbacks of a hybrid cloud are that knowledge of both the existing on-premises environment and the new cloud environment must be present and maintained, and there is a combination of pay-as-you-go costs in the cloud and investment and licensing costs in the on-premises environment.

8

AUTOMATION

8.1 Introduction

Until a few years ago, most servers, storage, and networks were configured manually. Systems managers installed hardware in racks, installed operating systems from installation media, added libraries and applications, patched the system with the latest software versions, and configured the software for that specific installation. However, this approach is slow, error-prone, and not easily repeatable. It introduces variations in server configurations that should be the same and makes the infrastructure very difficult to maintain.

An alternative is to automatically create and configure servers, storage, and networking, a concept known as infrastructure as code.

8.2 Infrastructure as code

Infrastructure as Code (IaC) is a way to deploy and manage infrastructure components based on a programming language, similar to how software developers write code to create applications.

IaC tools allow developers to define the desired state of their infrastructure in a programming language, which is then used to provision and manage infrastructure resources. This approach ensures that the infrastructure is consistent and can be deployed in a repeatable manner.

With IaC, deployment speeds are increased and greater consistency and reliability are achieved with fewer errors.

By treating infrastructure like code, organizations can automate the management of their infrastructure, enabling them to respond more quickly to changing business needs and reduce the risk of human error.

8.2.1 Declarative vs imperative languages

Computer code can either be declarative or imperative in nature.

- **Imperative** programming describes how to achieve a certain result by defining a sequence of steps. The focus is on how the program should accomplish a task.
- **Declarative** programming describes what the program should accomplish without defining the exact steps to get it done. Declarative programming languages focus on the end result, rather than how to achieve it.

Unlike most ordinary programming languages, such as C, Java and Python, most IaC languages are declarative. The IaC code describes what the infrastructure should look like, and executing the code deploys the infrastructure as described. This makes the code an important part of infrastructure documentation. It allows all systems managers to read how the entire infrastructure is put together.

If a change is made to the code, for example when a new VM is inserted into the code base, the IaC tool will check the state of the running infrastructure and compare it to the desired state as described in the code. In this example, it will determine that an additional VM is needed and will deploy only that new VM.

8.2.2 Versioning

To keep track of changes to software code over time, developers use version control systems. In these systems, files of software code are stored in repositories. A repository automatically creates a new version of the code when code is pushed to the repository. All previous versions remain available, enabling the retrieval of a previous version of a file if necessary. The repository is generally used by multiple developers, with each developer writing or maintaining part of the code.

IaC also benefits from version control systems. Git, GitHub and GitLab are the most widely used tools for version control.

- **Git** is a distributed version control system with a standalone command line interface tool, that provides features like branching, merging, and committing changes to code.

- **GitHub** is a web-based platform for hosting Git repositories. It provides a graphical interface for creating, managing, and sharing Git repositories.
- **GitLab** provides features similar to GitHub, but can be self-hosted.

8.2.3 Commonly used IaC languages

There are several commonly used IaC languages. Below are some of the most popular ones:

Terraform is a popular open-source tool and Domain-Specific Language (DSL) for building, changing, and versioning infrastructure. Terraform is cloud agnostic, which means that it has a generic syntax can be used to configure a wide range of cloud providers and infrastructure platforms, including AWS, Azure, GCP, Kubernetes, Red Hat OpenShift, databases like MySQL and PostgreSQL, firewalls, and more. But it must be noted that each platform needs its own configuration details – in Terraform, configuring an EC2 VM in AWS is done differently than configuring a VM in Azure.

SKIPPED TEXT

IS 2020.3 Curriculum reference matrix

The IS 2020 is a Competency Model for Undergraduate Programs in Information Systems from the Association for Computing Machinery (ACM). It contains several competence areas, including IT Infrastructure (competence area 3). IS 2020.3 based courses[39] offer an introduction to IT infrastructure topics for students majoring in Information Systems. It provides the students the knowledge and skills that they need for communicating effectively with professionals whose special focus is on hardware and systems software technology and for designing organizational processes and software solutions that require in-depth understanding of the IT infrastructure capabilities and limitations.

This book covers all topics that are part of the IS 2020.3 curriculum. The matrices in this appendix specify the relationship between the IS 2020.3 curriculum competences and the sections in this book.

Competency 1

Explain key infrastructure concepts, including how it functions, how to define critical functions, and how to plan and manage infrastructure.

Topic	Section in this book
Individual components of IT infrastructure	2, 2.5
Functions of IT infrastructure	All chapters 8, 9, 10, 11, 12, 13
Plan and manage IT infrastructure	14, 17, 18, 19, 20
Organizing structures and processes	14, 17, 18, 19, 20
Role of IT infrastructure in business	1.2, 2.6

Competency 2

Explain the principles of layered network architectures.

Topic	Section in this book
Layers of the TCP/IP protocol suite	9.3.4.1, 9.3.5.1
Layers of the OSI model	9.3.1
Duties of each layer of TCP/IP protocol suite	9.3.4.2, 9.3.4.3, 9.3.4.4, 9.3.4.5, 9.3.4.6, 9.3.5.2
Duties of each layer of OSI model	Par 9.3.2 to 9.3.8
Network security	9.7

Competency 3

Explain the components of IT infrastructure solutions from client/server, network hardware, (including wireless and wired).

Topic	Section in this book
Components of a network	9.5.1, 9.5.2
Components of Client/server	2.3
Wired networks	9.3.2.1, 9.3.2.2, 9.3.2.3, 9.3.2.4, 9.3.3.2
Wireless protocols	9.3.3.3, 9.3.3.6

Competency 4

Explain the principles of network software and configuration.

Topic	Section in this book
Configuration and setup processes on network hardware, software and other supporting devices and components	17
Four types of computer networks, LAN, WAN, PAN, MAN	9.3.3.1
Network topologies: Mesh, Star, Bus, Ring, Hybrid	9.2

Competency 5

Explain network protocols and their configuration.

Topic	Section in this book
Transmission Control Protocol (TCP)	9.3.5.1
Internet Protocol (IP)	9.3.4.1, 9.3.4.2, 9.3.4.3
User Datagram Protocol (UDP)	9.3.5.1
Post office Protocol (POP)	9.3.8.5
Simple mail transport Protocol (SMTP)	9.3.8.6
File Transfer Protocol (FTP)	9.3.8.7
HyperText Transfer Protocol (HTTP)	9.3.8.8
HyperText Transfer Protocol Secure (HTTPS)	9.3.8.8

Competency 6

Explain security principles as they pertain to networks.

Topic	Section in this book
Basic forms of system attacks	7.3, 7.1.2, 7.1.3, 7.1.3, 7.2
Access control to computers and networks	7.5.1.6
Techniques to make data secure	7.1.1, 9.7
Strengths and weaknesses of passwords	7.5.1.6
Basic features of cryptography	7.5.1.7
Firewalls and types of firewall protection	9.7.2
Techniques to secure wireless communication	9.3.3.3
Advantages of a security policy	7.5.1.1

Competency 7

Examine and critique IT infrastructure for organizations.

Topic	Section in this book
Infrastructure components	All chapters 8, 9, 10, 11, 12, 13
Infrastructure planning	14, 17, 18, 19, 20
Continuity planning	5.4.5

Competency 8

Examine and critique IT server architecture (both physical or cloud-based).

Topic	Section in this book
Server Components	11
Cloud configuration	3

Competency 9

Explain concepts of Enterprise Architecture.

Topic	Section in this book
Foundations of TOGAF	18.2.1
Foundations of ITIL	18.2.2

SKIPPED TEXT

End notes

[1] Box, G. E. P., and Draper, N. R., (1987), Empirical Model Building and Response Surfaces, John Wiley & Sons, New York, NY, p. 424

[2] http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[3] I would like to thank Gregory Short, instructor at Purdue University – Fort Wayne for his help improving this formula. The original formula only worked for systems that all have the same availability percentage.

[4] "Why Do Computers Stop and What Can Be Done About It?", Tandem Computers technical report 85.7, June 1985

[5] Source: https://berriprocess.com/en/bottleneck-law/

[6] http://www.seagate.com/www-content/product-content/enterprise-performance-savvio-fam/enterprise-performance-15k-hdd/ent-perf-15k-5/en-gb/docs/enterprise-performance-15k-hdd-ds1797-3-1406gb.pdf

[7] http://www.anandtech.com/show/6372/memory-performance-16gb-ddr31333-to-ddr32400-on-ivy-bridge-igp-with-gskill

[8] http://www.extremetech.com/extreme/188776-how-l1-and-l2-cpu-caches-work-and-why-theyre-an-essential-part-of-modern-chips/2

[9] https://www.cisecurity.org/cis-benchmarks/

[10] Copyright: ddgenome
Source: http://www.flickr.com/photos/ddgenome/3730193968/sizes/z/in/set-72157603633991423/

[11] Source: https://www.akibia.com/how-big-is-a-google-data-center/#:~:text=The%20Google%20data%20center%20is,environmentally%20responsible%20and%20energy%2Defficient

[12] Source: https://dgtlinfra.com/google-cloud-data-center-locations/

[13] Source: https://commons.wikimedia.org/wiki/File:Google_Data_Center,_Council_Bluffs_Iowa_%2849062863796%29.jpg

[14] Source: https://www.datacenterfrontier.com/design/article/11431484/inside-amazon8217s-cloud-computing-infrastructure

[15] Copyright: F. Dominec
Source: http://commons.wikimedia.org/wiki/File:Coaxial_cable_cut.jpg

[16] Copyright: Clive Darra.
Source: https://www.flickr.com/photos/osde-info/20943120629/

[17] Updated using https://businessquant.com/seagate-average-hdd-capacity-worldwide and https://www.techtarget.com/searchstorage/feature/Hard-disk-drives-to-remain-dominant-storage-media

[18] As published in Scientific American, July 2005
http://www.scientificamerican.com/article/kryders-law/

[19] Picture by Daniel Sancho.
Source: https://www.flickr.com/photos/teclasorg/2852716477/

[20] Public domain picture. Source: https://commons.wikimedia.org/wiki/File:ENIAC-changing_a_tube.jpg

[21] Copyright: Robert van Jemimus
Source: https://flic.kr/p/6SZom

[22] Copyright: Robert van Jemimus
Source: https://flic.kr/p/7zAdj

[23] Definition by Wikipedia:
https://en.wikipedia.org/wiki/Central_processing_unit

[24] Source: http://www.hardwarezone.com.sg/feature-intel-xeon-5130-and-5160-2-way-smp-performance-review

[25] Please read the excellent description on https://twitter.com/kenshirriff/status/1599120928050483207?s=61&t=nnNTht9oj5ALinG__JCNWg

[26] Source: https://www.intel.com/content/www/us/en/newsroom/news/13th-gen-core-launch.html#gs.oqrxyp

[27] Copyright: Luca Detomi
Source: http://en.wikipedia.org/wiki/File:Intel_4004.jpg

[28] Electronics, Volume 38, Number 8, April 19, 1965
http://www.cs.utexas.edu/~pingali/CS395T/2013fa/papers/moorespaper.pdf

[29] https://pubs.acs.org/doi/10.1021/j100785a001

[30] According to http://wiki.midrange.com/index.php/OS/400_101

[31] https://www.ibm.com/support/pages/node/668157

[32] Andrew S. Tanenbaum. Operating Systems: Design and Implementation. Prentice-Hall, 1987, ISBN 0-13-637406-9

[33] The original usenet post can be found here:
https://groups.google.com/forum/#!topic/comp.os.minix/dlNtH7RRrGA[1-25]

[34] Top500.org

[35] Copyright: picture by ArnoldReinhold - Own work, CC BY-SA 3.0
Source: https://commons.wikimedia.org/w/index.php?curid=31105488

[36] Copyright: Ruben de Rijcke
Source: http://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg

[37] https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202208-202209-bar

[38] According to Samsung:
http://www.storagevisions.com/2013/Book/Michael%20Willett.pdf

[39] For a full description of the curriculum, see https://www.acm.org/binaries/content/assets/education/curricula-recommendations/is2020.pdf