UNIT - 3
Data-Intensive Computing

RAID (Redundant Arrays of Independent Disks)

Why Data Redundancy?

Data redundancy, although taking up extra space, adds to disk reliability. This means, that in case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation. On the other hand, if the data is spread across multiple disks without the RAID technique, the loss of a single disk can affect the entire data.

RAID (Redundant Arrays of Independent Disks)

Different RAID Levels

1. RAID-0 (Stripping)
2. RAID-1 (Mirroring)
3. RAID-2 (Bit-Level Stripping with Dedicated Parity)
4. RAID-3 (Byte-Level Stripping with Dedicated Parity)
5. RAID-4 (Block-Level Stripping with Dedicated Parity)
6. RAID-5 (Block-Level Stripping with Distributed Parity)
7. RAID-6 (Block-Level Stripping with two Parity Bits)

RAID-0 (Stripping)

Blocks are “stripped” across disks.

- In the figure, blocks “0,1,2,3” form a stripe.
- Instead of placing just one block into a disk at a time, we can work with two (or more) blocks placed into a disk before moving on to the next one.

RAID 0

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

RAID-0 (Stripping)

Advantages

It is easy to implement.

It utilizes the storage capacity in a better way.

Disadvantages

A single drive loss can result in the complete failure of the system.

Not a good choice for a critical system.

RAID-1 (Mirroring)

More than one copy of each block is stored in a separate disk. Thus, every block has two (or more) copies, lying on different disks.

RAID 1

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

The above figure shows a RAID-1 system with mirroring level 2.
RAID 0 was unable to tolerate any disk failure. But RAID 1 is capable of reliability

RAID-1 (Mirroring)

Advantages

It covers complete redundancy.

It can increase data security and speed.

Disadvantages

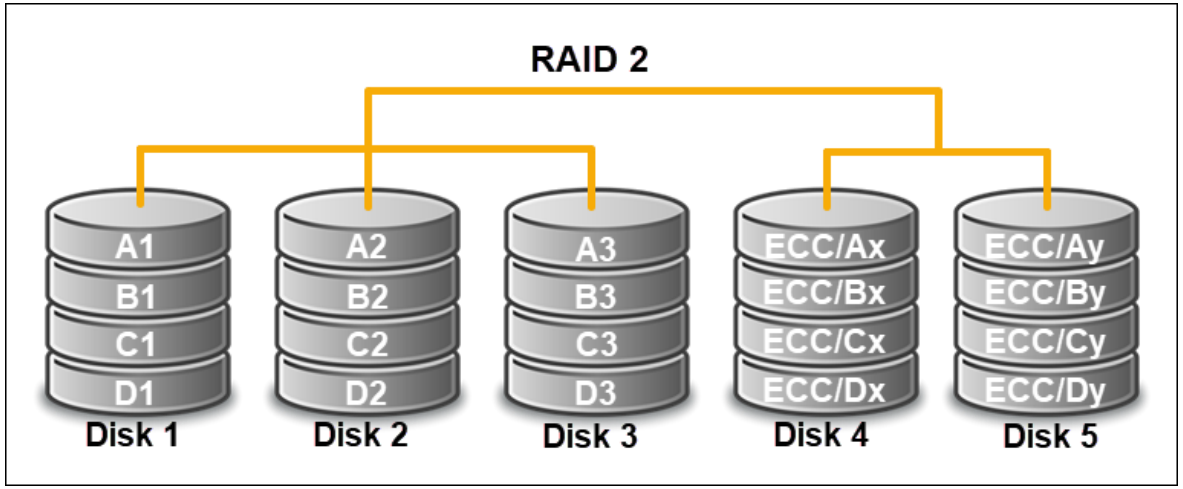
It is highly expensive.

Storage capacity is less.

RAID-2 (Bit-Level Stripping with Dedicated Parity)

In Raid-2, the error of the data is checked at every bit level. Here, we use Hamming Code Parity Method to find the error in the data.

- It uses one designated drive to store parity.
- The structure of Raid-2 is very complex as we use two disks in this technique. One word is used to store bits of each word and another word is used to store error code correction.
- It is not commonly used.



RAID-2 (Bit-Level Stripping with Dedicated Parity)

Advantages

1.In case of Error Correction, it uses hamming code.
2.It Uses one designated drive to store parity.

Disadvantages

1.It has a complex structure and high cost due to extra drive.
2.It requires an extra drive for error detection.

RAID-3 (Byte-Level Stripping with Dedicated Parity)

It consists of byte-level striping with dedicated parity striping.

- At this level, we store parity information in a disc section and write to a dedicated parity drive.
- Whenever failure of the drive occurs, it helps in accessing the parity drive, through which we can reconstruct the data.

RAID 3

Disk 0	Disk 1	Disk 2	Disk 3
15	16	17	P(15, 16, 17)
18	19	20	P(18, 19, 20)
21	22	23	P(21, 22, 23)
24	25	26	P(24, 25, 26)

RAID-3 (Byte-Level Stripping with Dedicated Parity)

Advantages

Data can be transferred in bulk.

Data can be accessed in parallel.

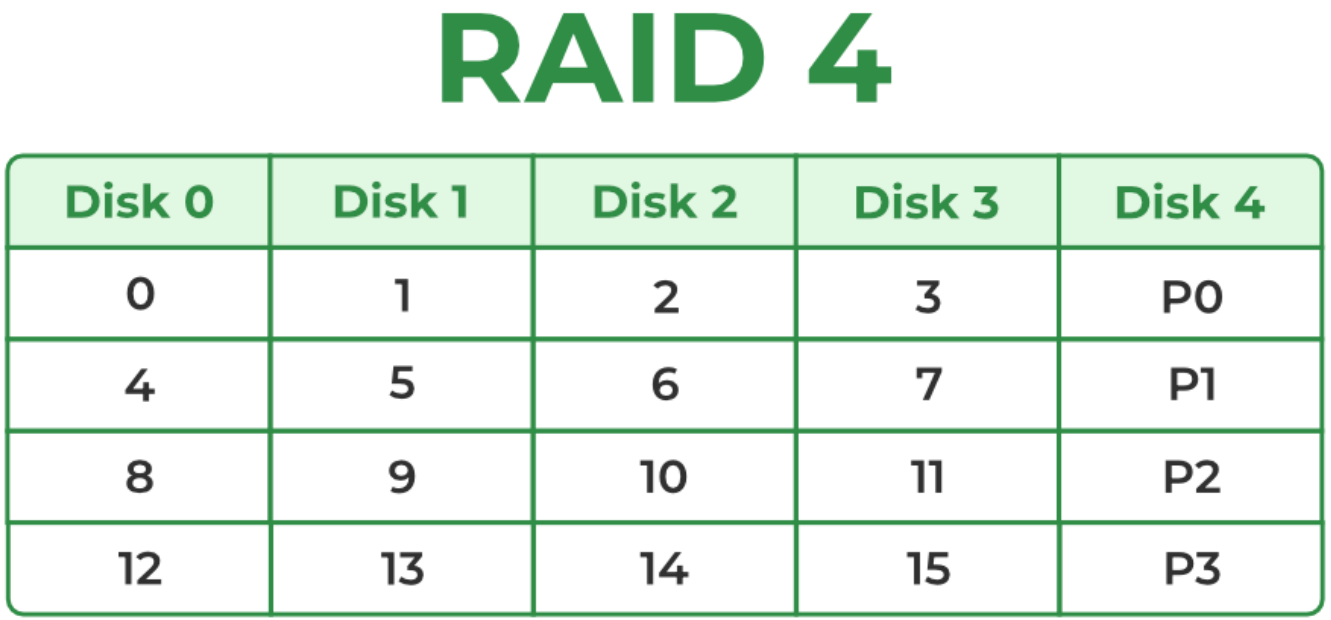
Disadvantages

It requires an additional drive for parity.

In the case of small-size files, it performs slowly.

RAID-4 (Block-Level Stripping with Dedicated Parity)

Instead of duplicating data, this adopts a parity-based approach.



•In the figure, we can observe one column (disk) dedicated to parity.
•Parity is calculated using a simple XOR function. If the data bits are 0,0,0,1 the parity bit is XOR(0,0,0,1) = 1. If the data bits are 0,1,1,0 the parity bit is XOR(0,1,1,0) = 0. A simple approach is that an even number of ones results in parity 0, and an odd number of ones results in parity 1.

RAID-4 (Block-Level Stripping with Dedicated Parity)

Advantages

It helps in reconstructing the data if at most one data is lost.

Disadvantages

It can’t help in reconstructing when more than one data is lost.

RAID-5 (Block-Level Stripping with Distributed Parity)

This is a slight modification of the RAID-4 system where the only difference is that the parity rotates among the drives.

RAID 5

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Advantages

1.Data can be reconstructed using parity bits.
2.It makes the performance better.

Disadvantages

1.Its technology is complex and extra space is required.
2.If both discs get damaged, data will be lost forever.

RAID-6 (Block-Level Stripping with two Parity Bits)

- Raid-6 helps when there is more than one disk failure. A pair of independent parities are generated and stored on multiple disks at this level. Ideally, you need four disk drives for this level.
- There are also hybrid RAIDs, which make use of more than one RAID level nested one after the other, to fulfill specific requirements.

RAID 6

Disk 1	Disk 2	Disk 3	Disk 4
A1	B1	P(B1)	P(B1)
A2	P(B2)	P(B2)	B2
P(B3)	P(B3)	A3	B3
P(B4)	A4	A4	P(B4)

RAID-6 (Block-Level Stripping with two Parity Bits)

Advantages

Very high data Accessibility.

Fast read data transactions.

Disadvantages

Due to double parity, it has slow write data transactions.

Extra space is required.

Advantages of RAID

Data redundancy: By keeping numerous copies of the data on many disks, RAID can shield data from disk failures.

Performance enhancement: RAID can enhance performance by distributing data over several drives, enabling the simultaneous execution of several read/write operations.

Scalability: RAID is scalable, therefore by adding more disks to the array, the storage capacity may be expanded.

Versatility: RAID is applicable to a wide range of devices, such as workstations, servers, and personal PCs

Disadvantages of RAID

Cost: RAID implementation can be costly, particularly for arrays with large capacities.

Complexity: The setup and management of RAID might be challenging.

Decreased performance: The parity calculations necessary for some RAID configurations, including RAID 5 and RAID 6, may result in a decrease in speed.

Single point of failure: RAID is not a comprehensive backup solution, while offering data redundancy. The array's whole contents could be lost if the RAID controller malfunctions.

Data-intensive computing

Data-intensive computing focuses on a class of applications that deal with a large amount of data. Several application fields, ranging from computational science to social networking, produce large volumes of data that need to be efficiently stored, made accessible, indexed, and analyzed.

Data-intensive computing is concerned with **production**, **manipulation**, and **analysis** of large-scale data in the range of hundreds of megabytes (MB) to petabytes (PB) and beyond

Datasets are often maintained in repositories, which are infrastructures supporting the storage, retrieval, and indexing of large amounts of information

To facilitate the classification and search, relevant bits of information, called **metadata**, are attached to datasets

Challenges ahead

The huge amount of data produced, analyzed, or stored imposes requirements on the supporting infrastructures and middleware that are hardly found in the traditional solutions for distributed computing.

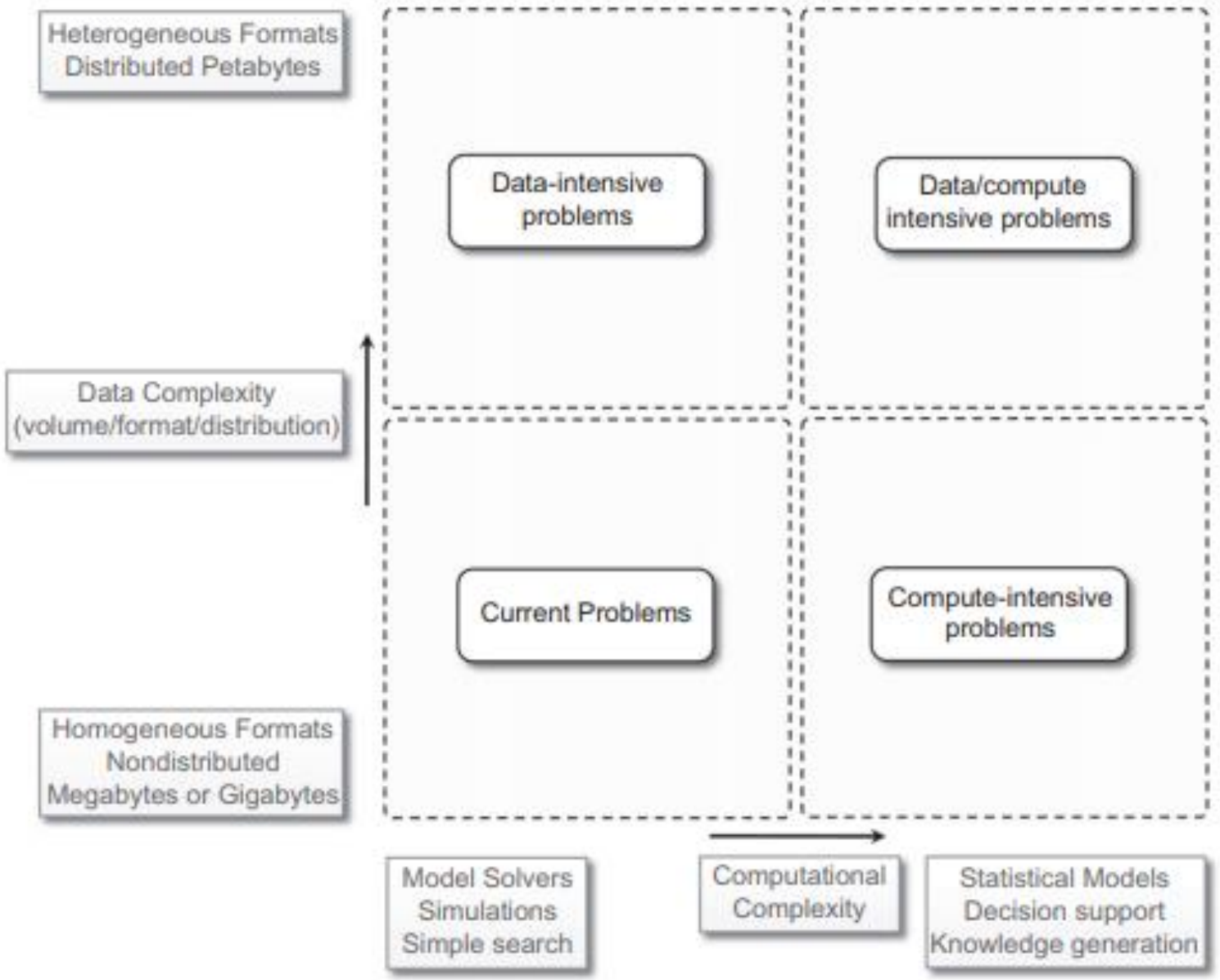
For example, the location of data is crucial as the need for moving terabytes of data becomes an obstacle for high-performing computations.

Data partitioning as well as **content replication** and **scalable algorithms** help in improving the performance of data-intensive applications

Data-Intensive Computing:

Open challenges in data-intensive computing are

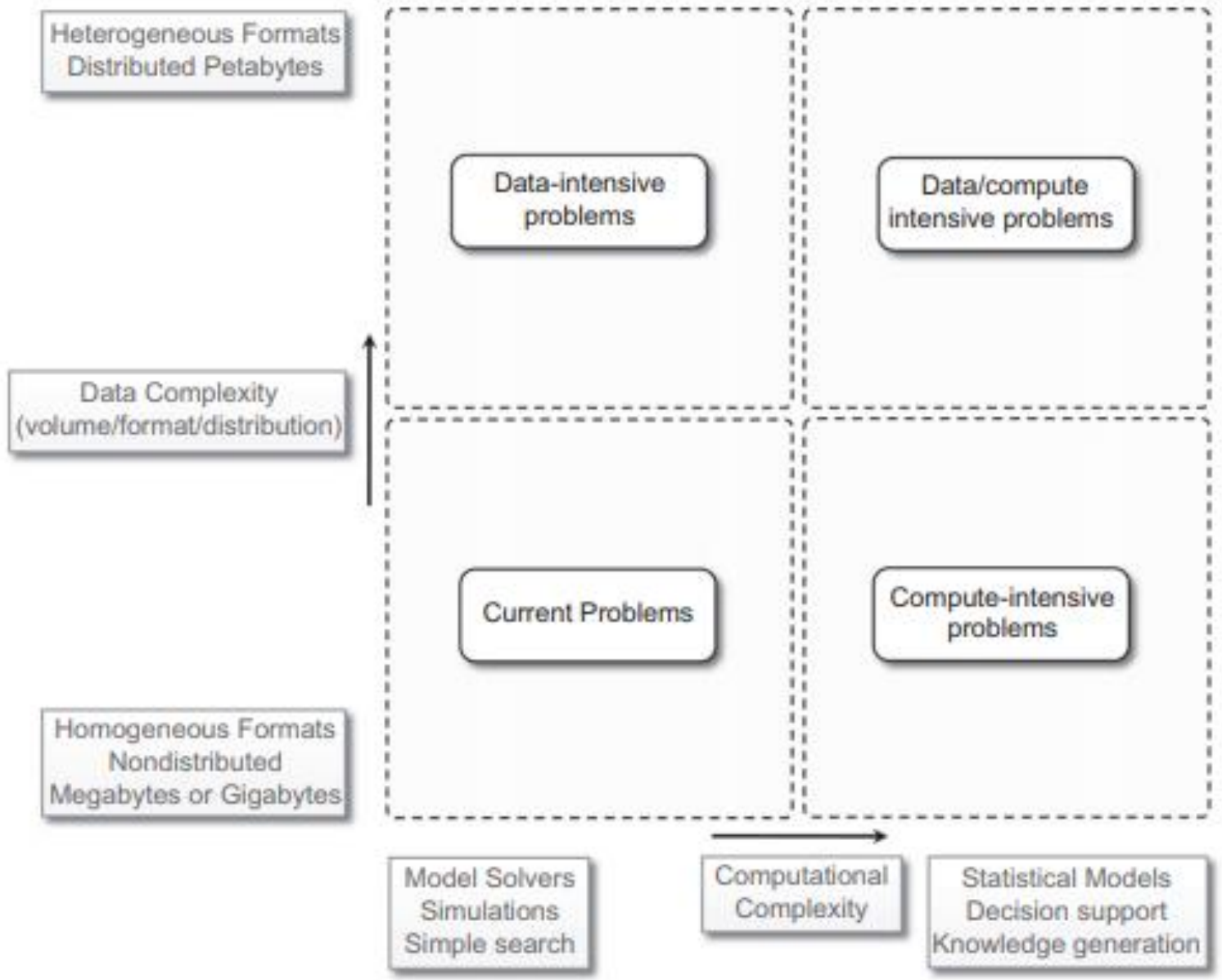
- Scalable algorithms that can search and process massive datasets
- New metadata management technologies that can scale to handle complex, heterogeneous, and distributed data sources
- Advances in high-performance computing platforms aimed at providing a better support for accessing in-memory multiterabyte data structures



Data-Intensive Computing:

Open challenges in data-intensive computing are

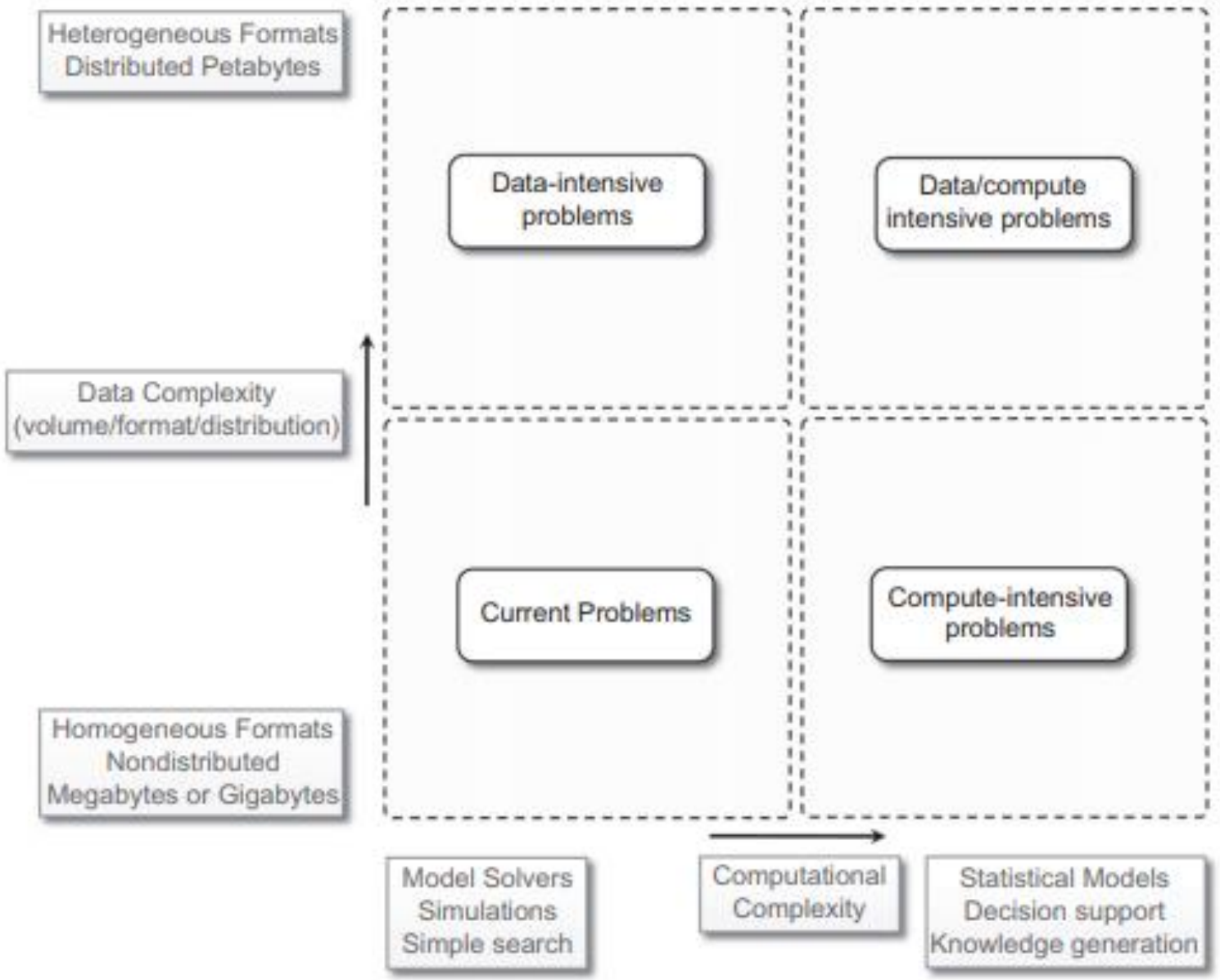
- High-performance, highly reliable, petascale distributed file systems
- Data signature-generation techniques for data reduction and rapid processing
- New approaches to software mobility for delivering algorithms that are able to move the computation to where the data are located



Data-Intensive Computing:

Open challenges in data-intensive computing are

- Specialized hybrid interconnection architectures that provide better support for filtering multi gigabyte data streams coming from high-speed networks and scientific instruments
- Flexible and high-performance software integration techniques that facilitate the combination of software modules running on different platforms to quickly form analytical pipelines



Data-Intensive Computing

Historical perspective

- The early age: high-speed wide-area networking
- Data grids
- Data clouds and “Big Data”
- Databases and data-intensive computing

Historical perspective

1. The early age: high-speed wide-area networking

- The evolution of technologies, protocols, and algorithms for data transmission and streaming has been an enabler of data-intensive computations
- In 1989, the first experiments in high-speed networking as a support for remote visualization of scientific data led the way
- The first data-intensive environment is reported to be the **MAGIC project**, a **DARPA-funded** collaboration working on distributed applications in large-scale, high-speed networks. Within this context, the **Distributed Parallel Storage System** (DPSS) was developed

Historical perspective

2. Data grids

Grid computing huge computational power and storage facilities could be obtained by harnessing heterogeneous resources across different administrative domains

A data grid provides services that help users **discover**, **transfer**, and **manipulate** large datasets stored in distributed **repositories** as well as create and manage copies of them.

Data grids offer two main functionalities:

- High-performance and reliable file transfer for moving large amounts of data
- Scalable replica discovery and management mechanisms for easy access to distributed datasets

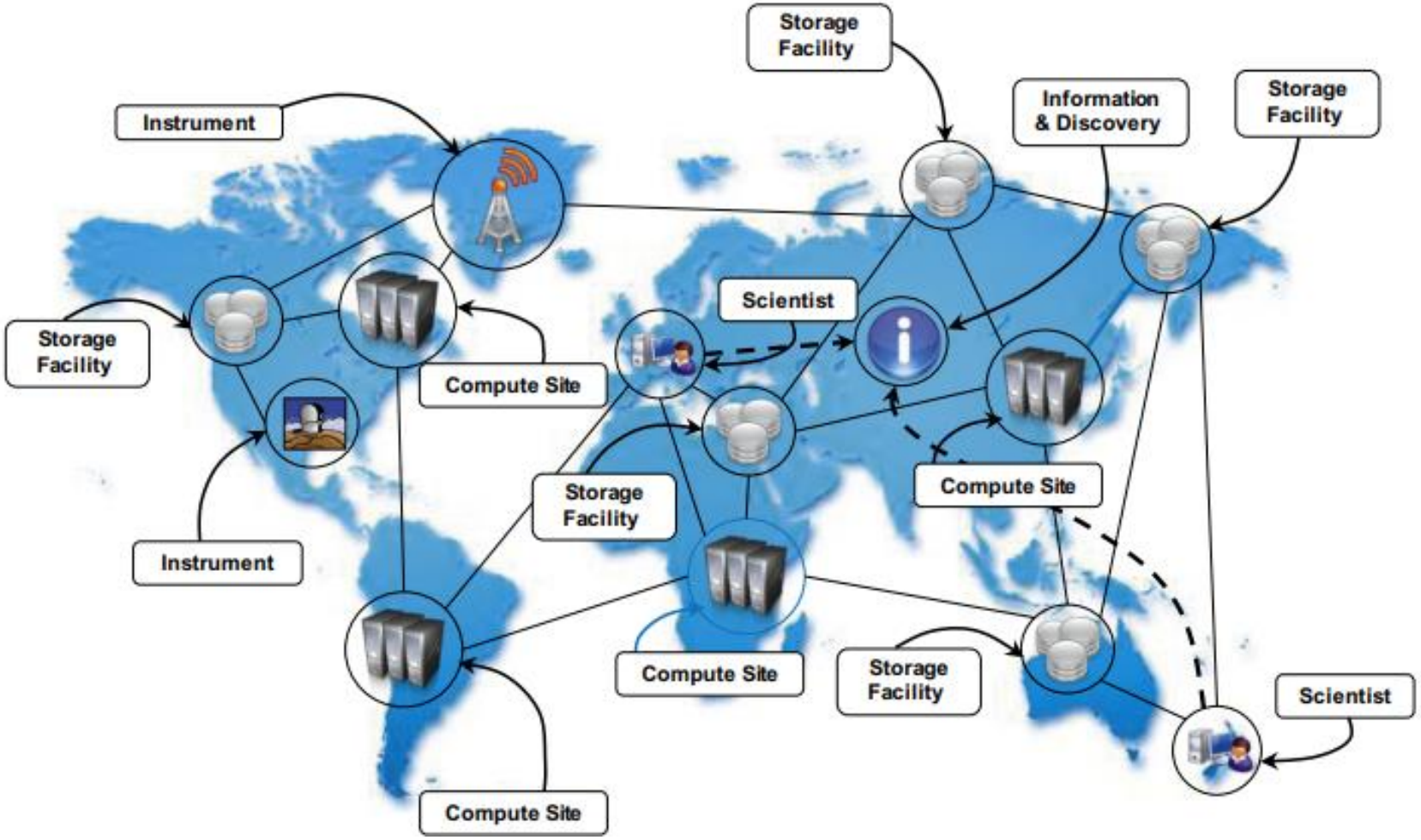
Historical perspective

2. Data grids

Data grids mostly provide storage and dataset management facilities as support for scientific experiments that produce huge volumes of data

For Example

The information, which can be locally processed, is then stored in repositories and made available for experiments and analysis to scientists



Data Grids characteristics and introduce new challenges

Massive datasets. The size of datasets can easily be on the scale of gigabytes, terabytes, and beyond. It is therefore necessary to minimize latencies during bulk transfers, replicate content with appropriate strategies, and manage storage resources.

Shared data collections. Resource sharing includes distributed collections of data. For example, repositories can be used to both store and read data.

Unified namespace. Data grids impose a unified logical namespace where to locate data collections and resources. Every data element has a single logical name, which is eventually mapped to different physical filenames for the purpose of replication and accessibility.

Access restrictions. Even though one of the purposes of data grids is to facilitate sharing of results and data for experiments, some users might want to ensure confidentiality for their data and restrict access to them to their collaborators. Authentication and authorization in data grids involve both coarse-grained and fine-grained access control over shared data collections.

Data clouds and "Big Data"

Large datasets have mostly been the domain of scientific computing. In this massive amounts of data are being produced, mined, and crunched by companies that provide Internet services such as searching, online advertising, and social media

It is critical for such companies to efficiently analyze these huge datasets because they constitute a precious source of information about their customers.

Log analysis is an example of a **data-intensive** operation that is commonly performed in this context; companies such as Google have a massive amount of data in the form of logs that are daily processed using their **distributed infrastructure.**

The term Big Data applies to datasets of which the size is beyond the ability of commonly used software tools to capture, manage, and process within a tolerable elapsed time.

Data-Intensive Computing

Data clouds and "Big Data"

Cloud technologies support data-intensive computing in several ways:

- By providing a large amount of compute instances on demand, which can be used to process and analyze large datasets in parallel.
- By providing a storage system optimized for keeping large blobs of data and other distributed data store architectures.
- By providing frameworks and programming APIs optimized for the processing and management of large amounts of data. These APIs are mostly coupled with a specific storage infrastructure to optimize the overall performance of the system.

Databases and data-intensive computing

Traditionally, distributed databases have been considered the natural evolution of database management systems as the scale of the datasets becomes **unmanageable** with a single system.

Distributed databases are a collection of data stored at different sites of a computer network.

A distributed database can be created by splitting and scattering the data of an existing database over different sites or by federating together multiple existing databases.

Technologies for data-intensive computing

Data-intensive computing concerns the development of applications that are mainly focused on processing large quantities of data.

Models constitute a natural classification of the technologies supporting data-intensive computing.

1. Storage systems
2. Programming platforms

Storage systems

Due to the explosion of unstructured data in the form of blogs, Web pages, software logs, and sensor readings, the relational model in its original formulation does not seem to be the preferred solution for supporting data analytics on a large scale

Some factors contributing to this change are:

- Growing of popularity of Big Data
- Growing importance of data analytics in the business chain
- Presence of data in several forms, not only structured
- New approaches and technologies for computing

Storage systems

High-performance distributed file systems and storage clouds

- Distributed file systems constitute the primary support for data management.
- provide an interface whereby to store information in the form of files and later access them for read and write operations.

Most Common Distributed file system formats are

- Lustre
- IBM General Parallel File System (GPFS)
- Google File System (GFS
- Amazon Simple Storage Service (S3)

Storage systems

High-performance distributed file systems and storage clouds

1. Lustre

The Lustre file system is a massively parallel distributed file system that covers the needs of a small workgroup of clusters to a large-scale computing cluster.

Lustre is designed to provide access to **petabytes (PBs)** of storage to serve thousands of clients with an I/O throughput of **hundreds of gigabytes per second (GB/s).**

The system is composed of a **metadata server** that contains the **metadata about the file system** and a collection of object **storage servers** that are in charge of providing storage.

The file system implements a **robust failover strategy** and **recovery mechanism**, **making server failures** and recoveries transparent to clients.

Storage systems

High-performance distributed file systems and storage clouds

2. IBM General Parallel File System (GPFS)

- GPFS is a multiplatform distributed file system built over several years of academic research and provides advanced recovery mechanisms
- GPFS is built on the concept of shared disks, in which a collection of disks is attached to the file system nodes by means of some switching fabric
- GPFS distributes the metadata of the entire file system and provides transparent access to it, thus eliminating a single point of failure.

Storage systems

High-performance distributed file systems and storage clouds

2. Google File System (GFS)

Public Cloud Infrastructures

Public Cloud Infrastructures

1. Amazon Web Services
2. Google AppEngine
3. Microsoft Azure

Public Cloud Infrastructures

Public Cloud Infrastructures

Some Example Cloud Computing Offerings		
Vendor/Product	**Service Type**	**Description**
Amazon Web Services	IaaS, PaaS, SaaS	Amazon Web Services (AWS) is a collection of Web services that provides developers with compute, storage, and more advanced services. AWS is mostly popular for IaaS services and primarily for its elastic compute service EC2.
Google AppEngine	PaaS	Google AppEngine is a distributed and scalable runtime for developing scalable Web applications based on Java and Python runtime environments. These are enriched with access to services that simplify the development of applications in a scalable manner.
Microsoft Azure	PaaS	Microsoft Azure is a cloud operating system that provides services for developing scalable applications based on the proprietary Hyper-V virtualization technology and the .NET framework.
SalesForce.com and Force.com	SaaS, PaaS	SalesForce.com is a Software-as-a-Service solution that allows prototyping of CRM applications. It leverages the Force.com platform, which is made available for developing new components and capabilities for CRM applications.
Heroku	PaaS	Heroku is a scalable runtime environment for building applications based on Ruby.
RightScale	IaaS	Rightscale is a cloud management platform with a single dashboard to manage public and hybrid clouds.

Public Cloud Infrastructures

Amazon web services

- Amazon Web Services (AWS) is a platform that allows the development of flexible applications by providing solutions for elastic infrastructure scalability, messaging, and data storage.
- The platform is accessible through SOAP or RESTful Web service interfaces and provides a Web-based console where users can handle administration and monitoring of the resources required, as well as their expenses computed on a pay-as-you-go basis.

Public Cloud Infrastructures

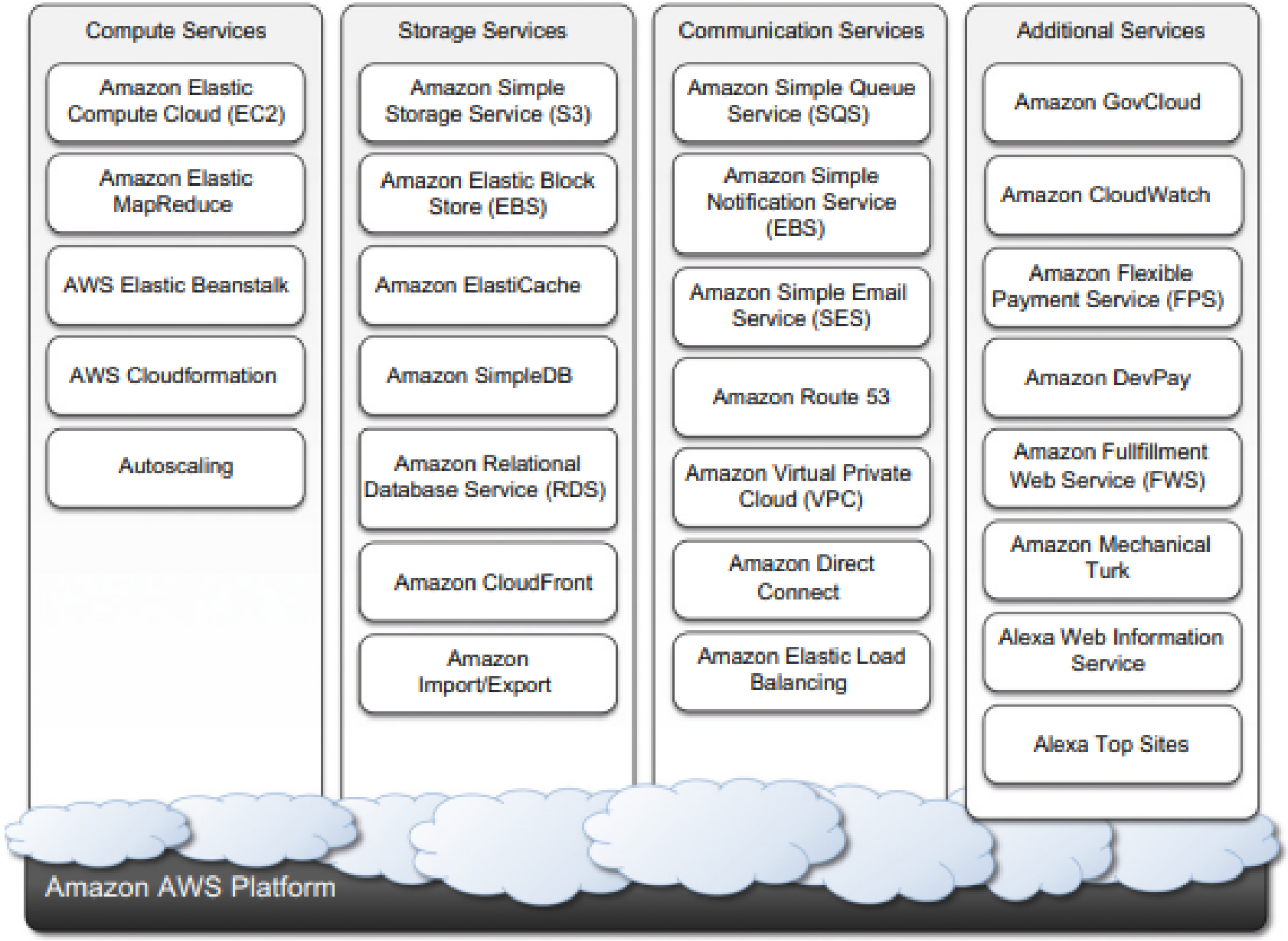
Amazon web services

the two most popular services of AWS are

Amazon Elastic Compute (EC2)
Amazon Simple Storage Service (S3).

Public Cloud Infrastructures

Amazon web services



Public Cloud Infrastructures

Compute services

- Compute services constitute the fundamental element of cloud computing systems.
- The fundamental service in this space is Amazon EC2, which delivers an IaaS solution that has served as a reference model for several offerings from other vendors in the same market segment

Amazon machine images

- Amazon Machine Images (AMIs) are templates from which it is possible to create a virtual machine.
- They are stored in Amazon S3 and identified by a unique identifier in the form of ami-xxxxxx and XML File

EC2 instances

- EC2 instances represent virtual machines. They are created using AMI as templates, which are specialized by selecting the number of cores, their computing power, and the installed memory.

Public Cloud Infrastructures

EC2 instances

- EC2 instances represent virtual machines. They are created using AMI as templates, which are specialized by selecting the number of cores, their computing power, and the installed memory.

Types of EC2 instances.

- **Standard instances.** This class offers a set of configurations that are suitable for most applications. EC2 provides three different categories of increasing computing power, storage, and memory.
- **Micro instances.** This class is suitable for those applications that consume a limited amount of computing power and memory and occasionally need bursts in CPU cycles to process surges in the workload. Micro instances can be used for small Web applications with limited traffic.
- **High-memory instances.** This class targets applications that need to process huge workloads and require large amounts of memory. Three-tier Web applications characterized by high traffic are the target profile. Three categories of increasing memory and CPU are available, with memory proportionally larger than computing power

Public Cloud Infrastructures

- **High-memory instances.** This class targets applications that need to process huge workloads and require large amounts of memory. Three-tier Web applications characterized by high traffic are the target profile. Three categories of increasing memory and CPU are available, with memory proportionally larger than computing power.
- **High-CPU instances.** This class targets compute-intensive applications. Two configurations are available where computing power proportionally increases more than memory.
- **Cluster Compute instances.** This class is used to provide virtual cluster services. Instances in this category are characterized by high CPU compute power and large memory and an extremely high I/O and network performance, which makes it suitable for HPC applications.
- **Cluster GPU instances.** This class provides instances featuring graphic processing units (GPUs) and high compute power, large memory, and extremely high I/O and network performance.

Public Cloud Infrastructures

EC2 environment

- EC2 instances are executed within a virtual environment, which provides them with the services they require to host applications.
- The EC2 environment is in charge of allocating addresses, attaching storage volumes, and configuring security in terms of access control and network connectivity.

Advanced compute services

- AWS CloudFormation (simple deployment mode with JSON formatted)
- AWS elastic beanstalk (Packaging and Deploying the application)
- Amazon elastic MapReduce (MapReduce , Connected to Hadoop)

Storage services

As the name suggests, S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface, which is quite similar to a distributed file system

Important differences that allow the infrastructure to be highly efficient:

The storage is organized in a two-level hierarchy. S3 organizes its storage space into buckets that cannot be further partitioned. This means that it is not possible to create directories or other kinds of physical groupings for objects stored in a bucket.

Stored objects cannot be manipulated like standard files. S3 has been designed to essentially provide storage for objects that will not change over time. Therefore, it does not allow renaming, modifying, or relocating an object. Once an object has been added to a bucket.

Content is not immediately available to users. The main design goal of S3 is to provide an eventually consistent data store. As a result, because it is a large distributed storage facility, changes are not immediately reflected

Storage services

As the name suggests, S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface, which is quite similar to a distributed file system

Important differences that allow the infrastructure to be highly efficient:

Requests will occasionally fail. Due to the large distributed infrastructure being managed, requests for object may occasionally fail. Under certain conditions, S3 can decide to drop a request by returning an internal server error

Storage services

Resource naming Buckets

objects, and attached metadata are made accessible through a REST interface. Therefore, they are represented by uniform resource identifiers (URIs) under the s3.amazonaws.com domain

- Canonical form: http://s3.amazonaws.com/bukect_name/
- Subdomain form: http://bucketname.s3.amazon.com/
- Virtual hosting form: http://bucket-name.com/

Storage services

Buckets

- A bucket is a container of objects. It can be thought of as a virtual drive hosted on the S3 distributed storage, which provides users with a flat store to which they can add objects.
- Buckets are toplevel elements of the S3 storage architecture and do not support nesting.

Objects and metadata

- Objects constitute the content elements stored in S3.
- Users either store files or push to the S3 text stream representing the object's content.
- An object is identified by a name that needs to be unique within the bucket in which the content is stored.

Storage services

Access control and security

Amazon S3 allows controlling the access to buckets and objects by means of Access Control Policies (ACPs).

READ allows the grantee to retrieve an object and its metadata and to list the content of a bucket as well as getting its metadata.

WRITE allows the grantee to add an object to a bucket as well as modify and remove it.

READ_ACP allows the grantee to read the ACP of a resource.

WRITE_ACP allows the grantee to modify the ACP of a resource.

FULL_CONTROL grants all of the preceding permissions

Storage services

Access control and security

Amazon S3 allows controlling the access to buckets and objects by means of Access Control Policies (ACPs).

READ allows the grantee to retrieve an object and its metadata and to list the content of a bucket as well as getting its metadata.

WRITE allows the grantee to add an object to a bucket as well as modify and remove it.

READ_ACP allows the grantee to read the ACP of a resource.

WRITE_ACP allows the grantee to modify the ACP of a resource.

FULL_CONTROL grants all of the preceding permissions

Google AppEngine

Google AppEngine is a PaaS implementation that provides services for developing and hosting scalable Web applications.

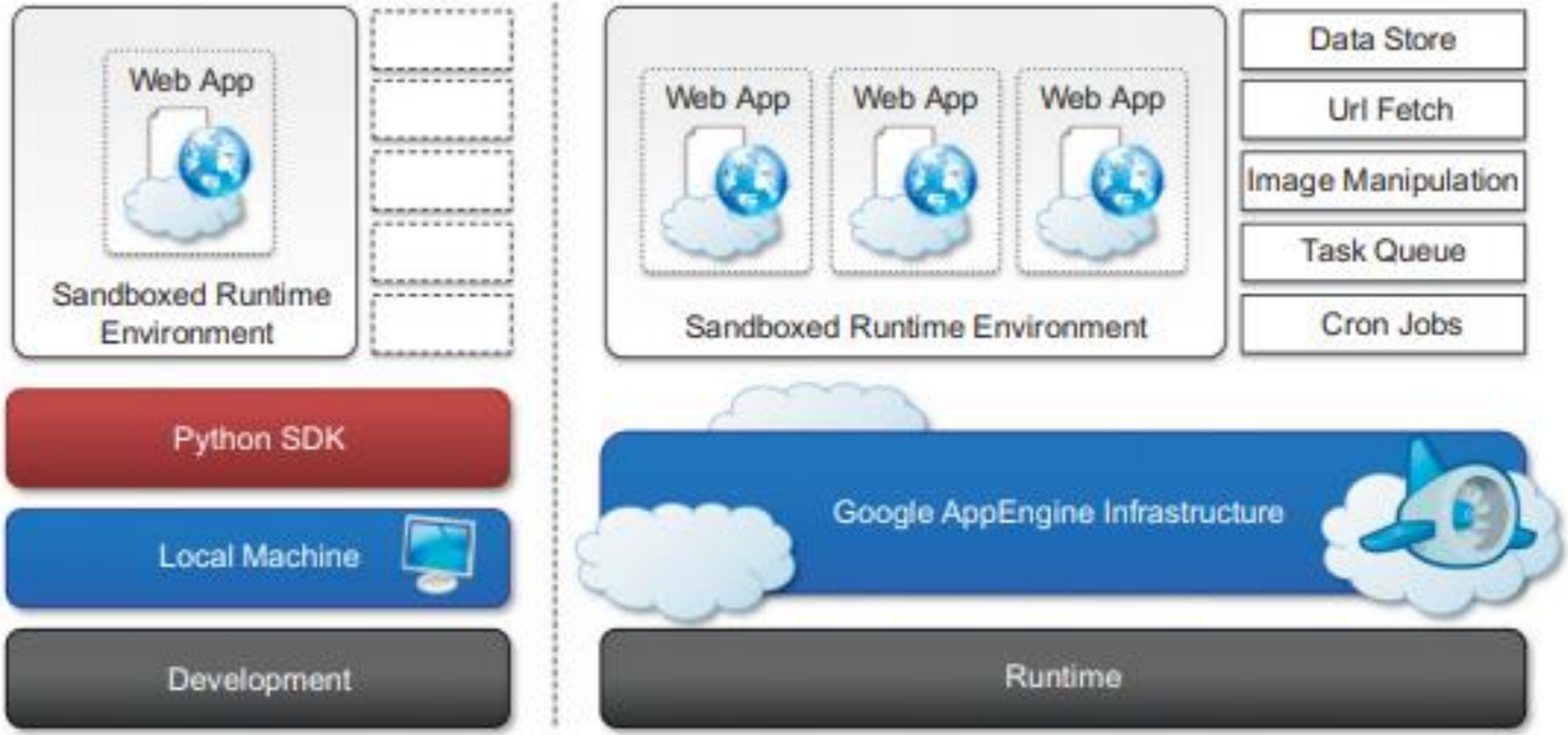
Architecture and core concepts

The platform is logically divided into four major components:

1. Infrastructure
2. Runtime environment
3. Underlying storage
4. Set of scalable services

Public Cloud Infrastructures

Google AppEngine Architecture



Google AppEngine Architecture

Infrastructure

- AppEngine hosts Web applications, and its primary function is to serve users requests efficiently. To do so, AppEngine's infrastructure takes advantage of many servers available within Google datacenters.
- For each HTTP request, AppEngine locates the servers hosting the application that processes the request, evaluates their load, and, if necessary, allocates additional resources (i.e., servers) or redirects the request to an existing server

Google AppEngine Architecture

Runtime environment

- The runtime environment represents the execution context of applications hosted on AppEngine.
- With reference to the AppEngine infrastructure code, which is always active and running, the runtime comes into existence when the request handler starts executing and terminates once the handler has completed.

Sandboxing

- One of the major responsibilities of the runtime environment is to provide the application environment with an isolated and protected context.
- AppEngine supports applications that are developed only with **managed** or **interpreted languages**, which by design require a runtime for **translating** their code into executable instructions.
- Therefore, sandboxing is achieved by means of **modified runtimes for applications** that disable some of the common features normally available with their default implementations.

Google AppEngine Architecture

Runtime environment

Supported runtimes

- Currently, it is possible to develop AppEngine applications using three different languages and related technologies: Java, Python, and Go
- AppEngine currently supports Java 6, and developers can use the common tools for Web application development in Java, such as the Java Server Pages (JSP)
- Support for Python is provided by an optimized Python 2.5.2 interpreter

Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data.

There are three different levels of storage: **in memory-cache**, **storage for semistructured data**, and **long-term storage for static data**.

Google AppEngine Architecture

Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data. There are three different levels of storage: in memory-cache, storage for semistructured data, and long-term storage for static data.

Static file servers Web applications are composed of dynamic and static data. Dynamic data are a result of the logic of the application and the interaction with the user. Static data often are mostly constituted of the components that define the graphical layout of the application (CSS files, plain HTML files, JavaScript files, images, icons, and sound files) or data files.

DataStore DataStore is a service that allows developers to store semistructured data. The service is designed to scale and optimized to quickly access data. DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key. Both the type of the key and the structure of the object can vary.

Google AppEngine Architecture

Application services

Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in Web applications: access to data, account management, integration of external resources, messaging and communication, image manipulation, and asynchronous computation.

UrlFetch Web 2.0 has introduced the concept of composite Web applications. Different resources are put together and organized as meshes within a single Web page

MemCache AppEngine provides developers with access to fast and reliable storage, which is DataStore. Despite this, the main objective of the service is to serve as a scalable and long-term storage, where data are persisted to disk redundantly in order to ensure reliability and availability of data against failures.

Google AppEngine Architecture

Application services

Account management Web applications often keep various data that customize their interaction with users. These data normally go under the user profile and are attached to an account. AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts

Image manipulation Web applications render pages with graphics. Often simple operations, such as adding watermarks or applying simple filters, are required. AppEngine allows applications to perform image resizing, rotation, mirroring, and enhancement by means of Image Manipulation, a service that is also used in other Google products.

Google AppEngine Architecture

Application life cycle

AppEngine provides support for almost all the phases characterizing the life cycle of an application: testing and development, deployment, and monitoring. The SDKs released by Google provide 338 CHAPTER 9 Cloud Platforms in Industry developers with most of the functionalities required by these tasks. Currently there are two SDKs available for development: Java SDK and Python SDK

Application development and testing

Application deployment and management

Google AppEngine Architecture

Cost model

AppEngine provides a free service with limited quotas that get reset every 24 hours. Once the application has been tested and tuned for AppEngine, it is possible to set up a billing account and obtain more allowance and be charged on a pay-per-use basis. This allows developers to identify the appropriate daily budget that they want to allocate for a given application.

Microsoft Azure Microsoft

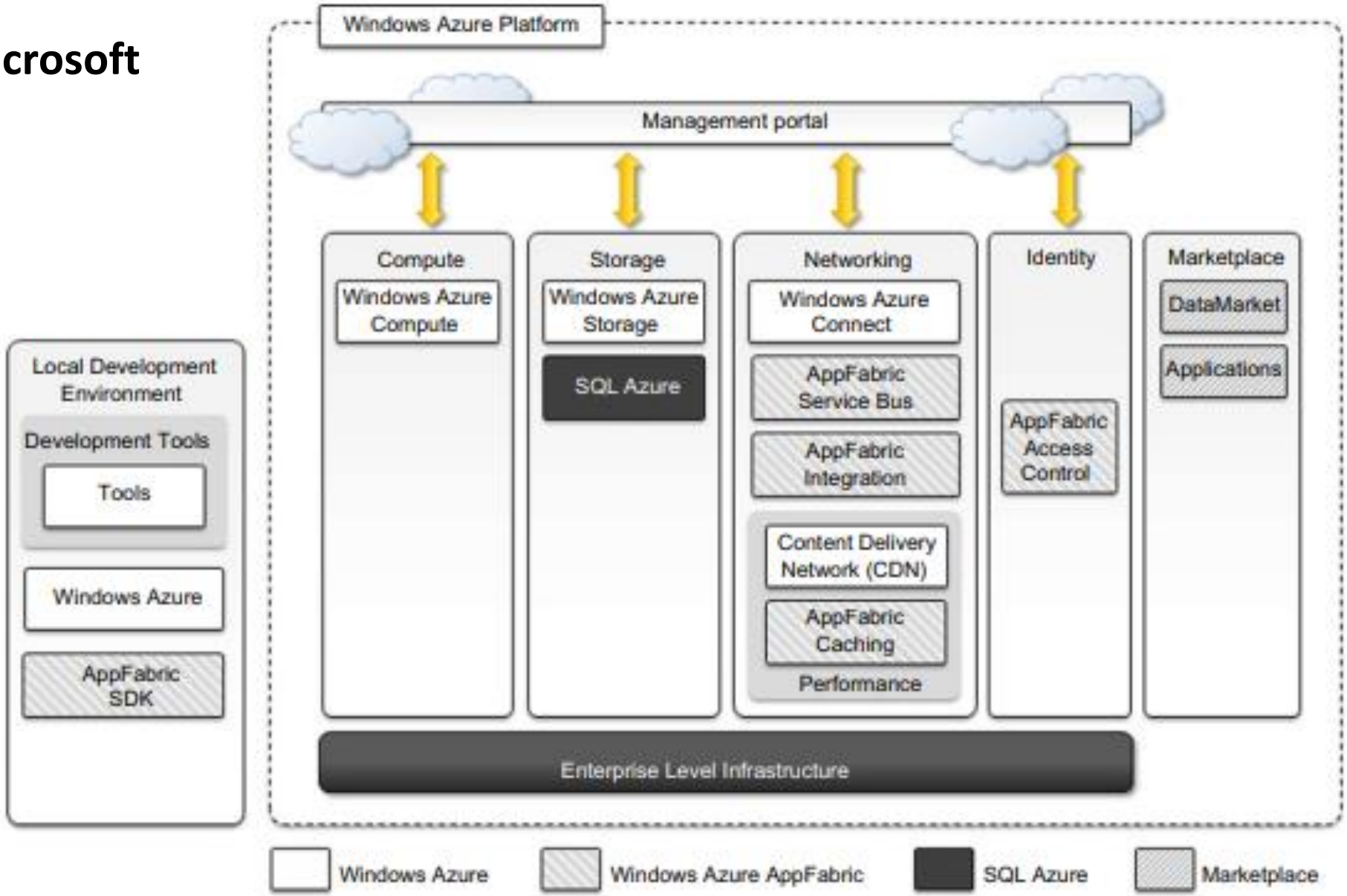
Windows Azure is a cloud operating system built on top of Microsoft datacenters' infrastructure and provides developers with a collection of services for building applications with cloud technology.

Services range from **compute**, **storage**, and **networking** to **application connectivity access control**, and **business intelligence**.

These services can be managed and controlled through the **Windows Azure Management Portal.**

Public Cloud Infrastructures

Microsoft Azure Microsoft



Microsoft Azure Microsoft

Compute services

Compute services are the core components of Microsoft Windows Azure, and they are delivered by means of the abstraction of roles.

A role is a runtime environment that is customized for a specific compute task.

Roles are managed by the Azure operating system and instantiated on demand in order to address surges in application demand.

Currently, there are three different roles:

Web role

Worker role

Virtual Machine (VM) role.

Microsoft Azure Microsoft

Compute services

Web role

- The Web role is designed to implement scalable Web applications.
- Web roles represent the units of deployment of Web applications within the Azure infrastructure. They are hosted on the **IIS 7 (Internet Information Server)** Web Server, which is a component of the infrastructure that supports Azure
- Visual Studio IDE is used to develop the web application, (ASP.NET, MVC Role, PhP, CG)

Worker role

- Worker roles are designed to host general compute services on Azure.
- The **Azure SDK** provides developers with convenient APIs and libraries that allow connecting the role with the service provided by the runtime and easily controlling its startup as well as being notified of changes in the hosting environment.

Microsoft Azure Microsoft

Compute services

- Virtual Machine (VM) role.
- The Virtual Machine role allows developers to fully control the computing stack of their compute service by defining a custom image of the Windows Server 2008 R2 operating system and all the service stack required by their applications

Microsoft Azure Microsoft

Storage services

Compute resources are equipped with local storage in the form of a directory on the local file system that can be used to temporarily store information that is useful for the current execution cycle of a role.

Blobs Azure allows storing large amount of data in the form of binary large objects (BLOBs) by means of the blobs service.

- **Block blobs.** Block blobs are composed of blocks and are optimized for sequential access; therefore they are appropriate for media streaming. Currently, blocks are of 4 MB, and a single block blob can reach 200 GB in dimension.
- **Page blobs.** Page blobs are made of pages that are identified by an offset from the beginning of the blob. A page blob can be split into multiple pages or constituted of a single page. This type of blob is optimized for random access and can be used to host data different from streaming. Currently, the maximum dimension of a page blob can be 1 TB.

Microsoft Azure Microsoft

Storage services

Azure drive

Page blobs can be used to store an entire file system in the form of a single Virtual Hard Drive (VHD) file.

Tables

Tables constitute a semistructured storage solution, allowing users to store information in the form of entities with a collection of properties

Queues

Queue storage allows applications to communicate by exchanging messages through durable queues, thus avoiding lost or unprocessed messages.