Big Data & Data Scientist

A BRIEF INTRODUCTION TO DATA SCIENCE AND BIG DATA

Business
Intelligence
Group

Matteo Golfarelli

Data Science & Data Scientist: a definition

Data Science is the extraction of knowledge from data. It employs techniques and theories drawn from many fields within the broad areas of statistics and information technology

The data scientist is an expert of such areas with a strong aptitude in understanding the business and the data. He is capable of transforming the information hidden in the data in a competitive advantage. His final goal is to create new business models in the so-called data-driven economy

Data Science is an evolution of Business Intelligence and it overcomes the following limitations:

Business Intelligence	Data Science
Works on Structured data only	Works on Structured and unstructured data
Requires a complex/rigid ETL process	Is based on a more flexible/agile/extemporary extraction process
Implemented by Computer Engineers Exploited by Business Users and data analysts	Implemented and exploited by Data Scientists and Data Enthusiasts
Implemented on Relational DBMS	Implemented on Big Data and NoSQL technologies

Why data science?

Business needs

Availability of large datasets

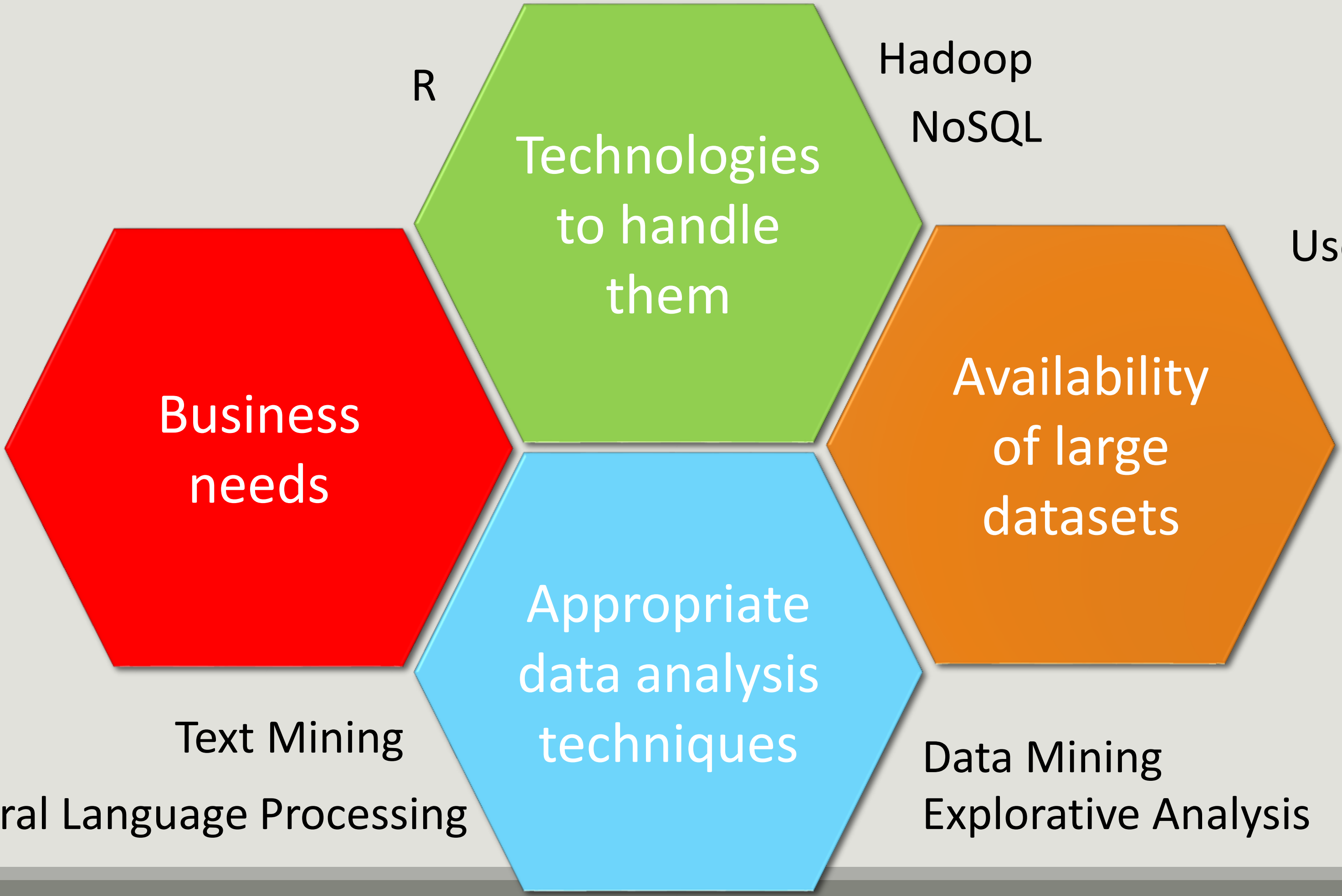
User Generated Contents

Internet of Things

Smart Cities

Smart Grid

Why data science?


Technologies to handle them
Business needs
Availability of large datasets
Appropriate data analysis techniques

R

Hadoop

NoSQL

User Generated Contents

Internet of Things

Smart City

Smart Grid

Text Mining

Natural Language Processing

Data Mining

Explorative Analysis

Where does Data science come from?

The enabler factors of this revolution are:

- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies

Where does Data science come from?

The enabler factors of this revolution are:

- Pressing user needs: users ask for quantitative information in the data-driven economy
- Data Availability: Internet of Things
- Computational power: Big Data Technologies

A personal experience that perfectly fits this concept is **Technogym** a worldwide leading company in the Wellness market

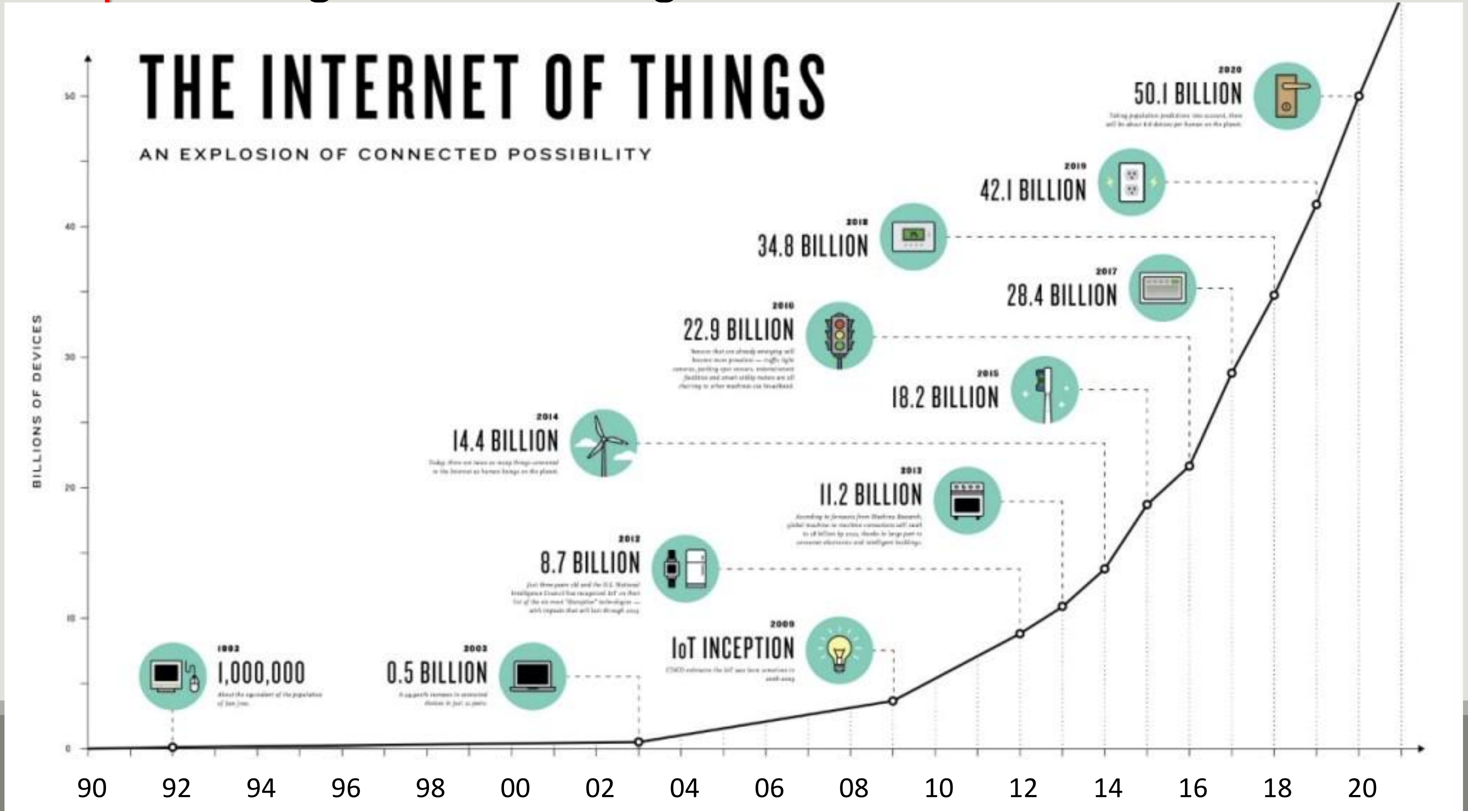
- Adds software and sensors to its machines that are stand-alone or connected at the Facility level
- Builds a social environment (social network, Apps) providing additional services to its customers
- **Moves its software framework to the cloud**

Bilions of data about each single Physical Activity Step is available for:
Customer profiling, Scientific analysis, Drop-Out prevention, etc.

Where does Data science come from?

The enabler factors of this revolution are:

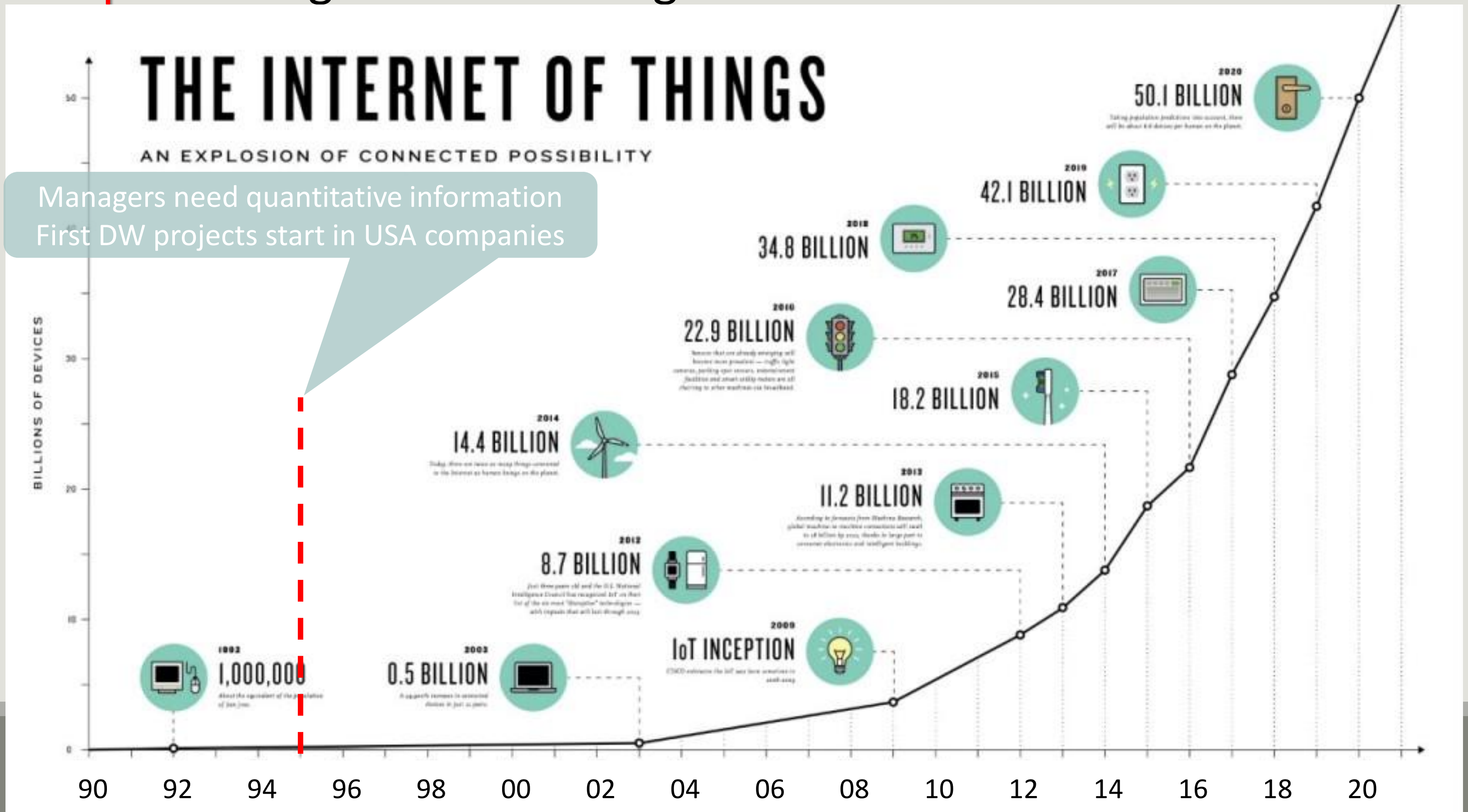
- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

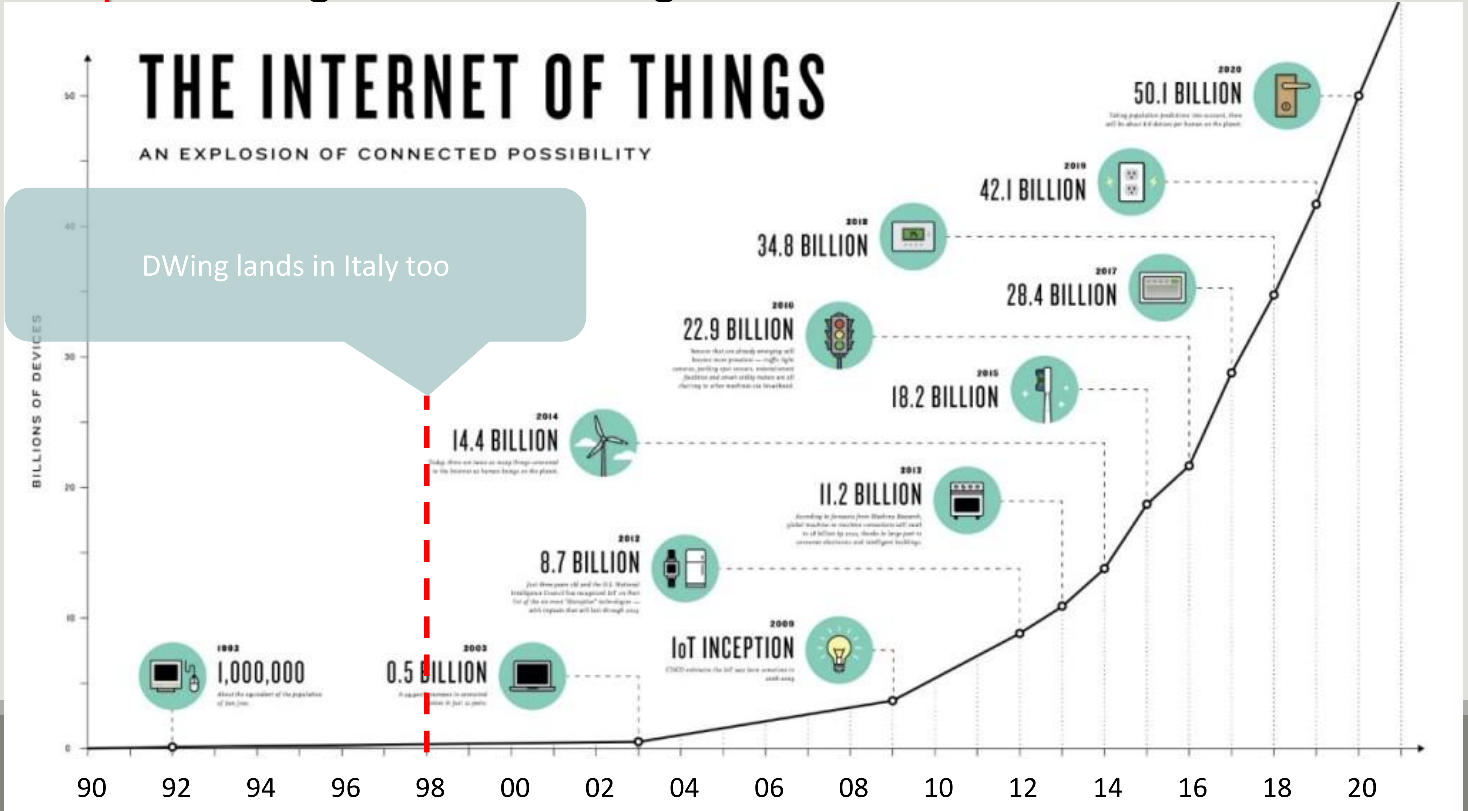
- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

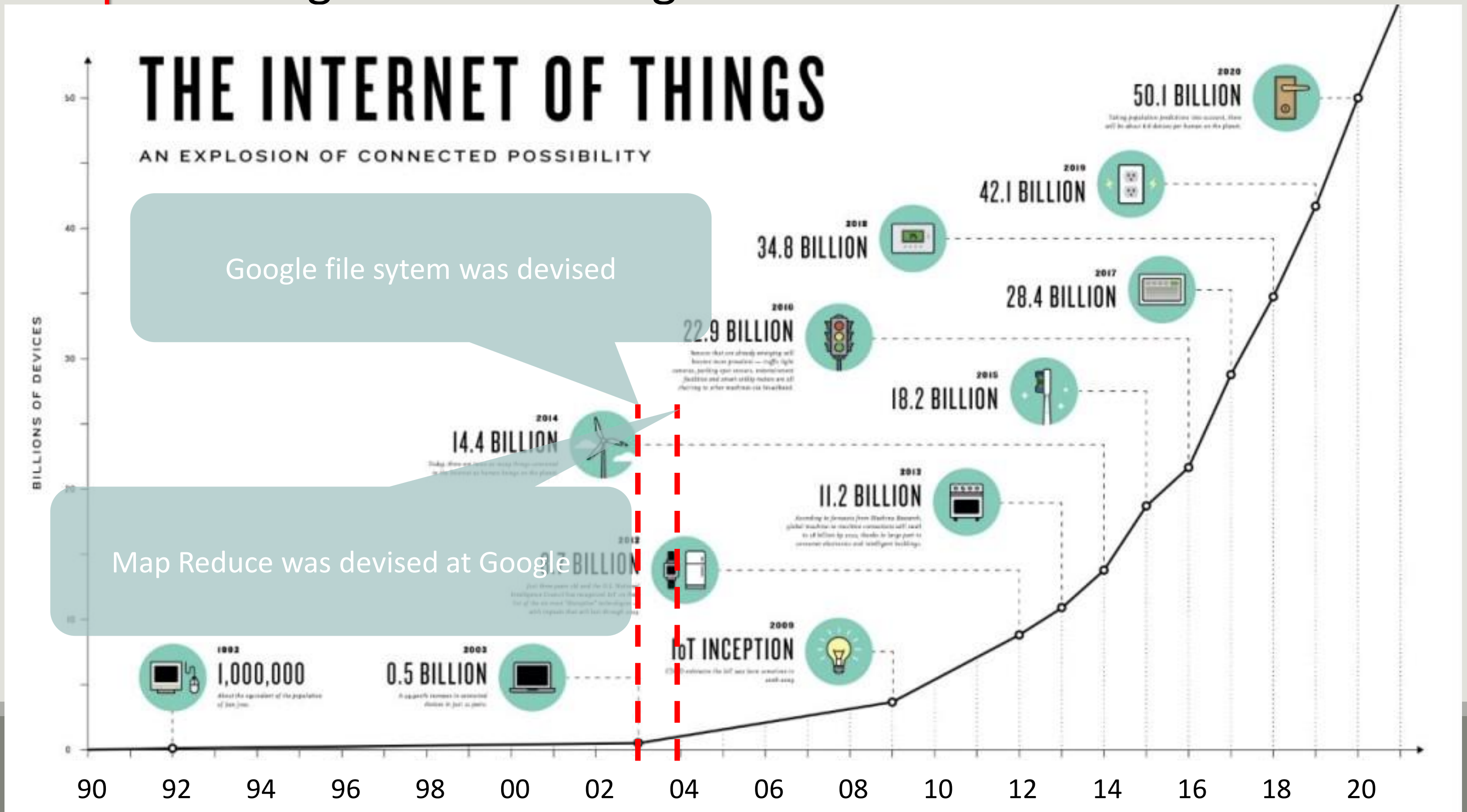
- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

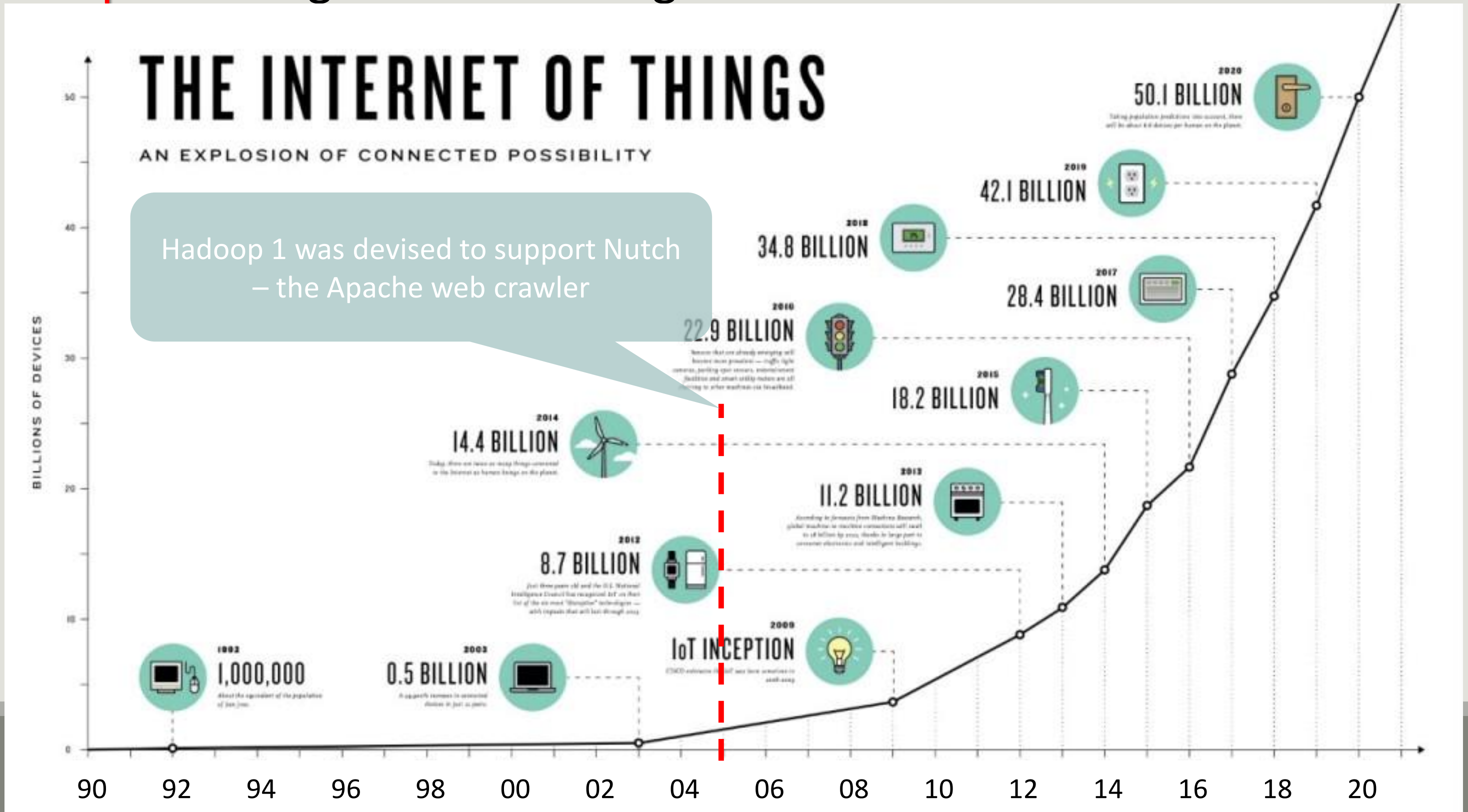
- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

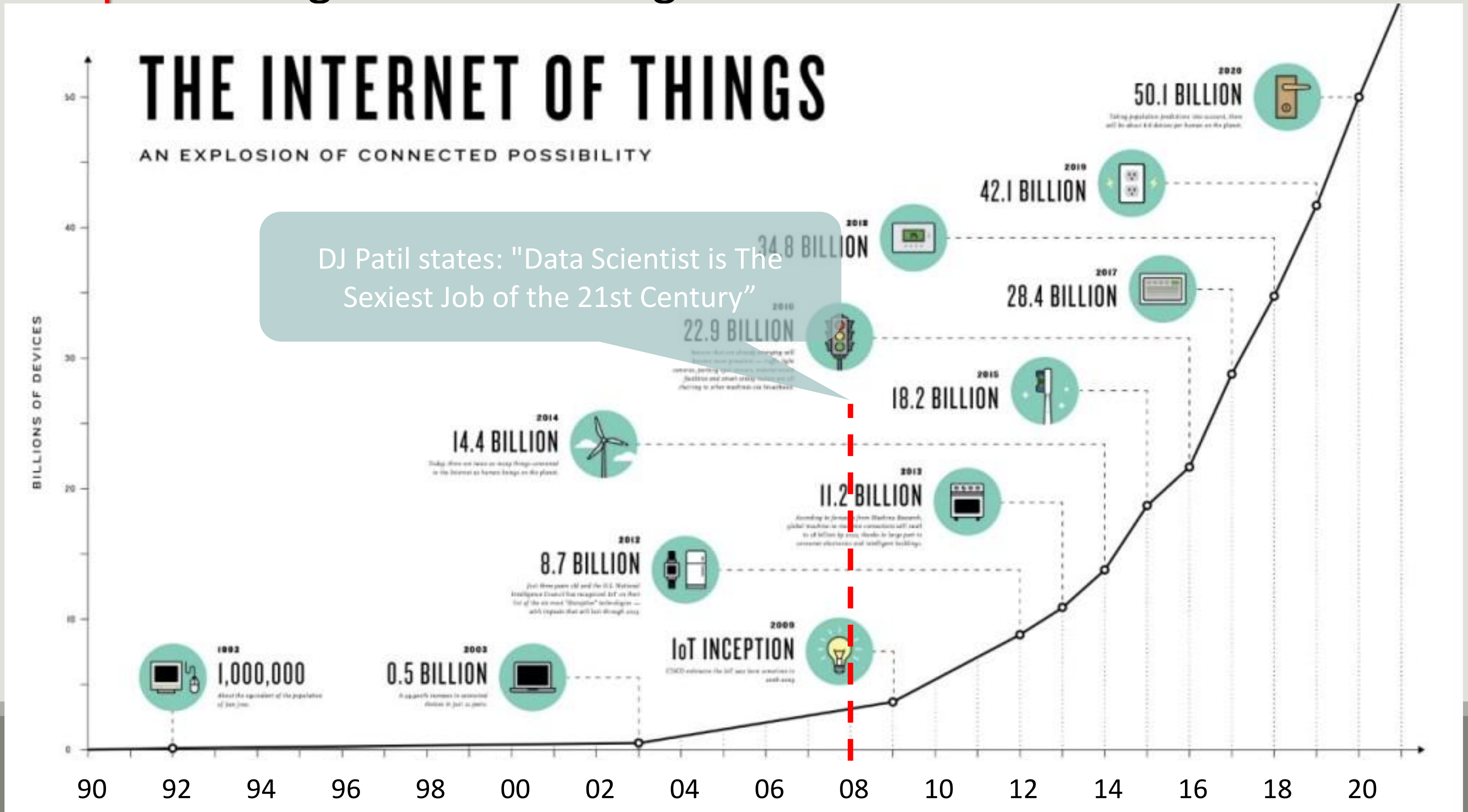
- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

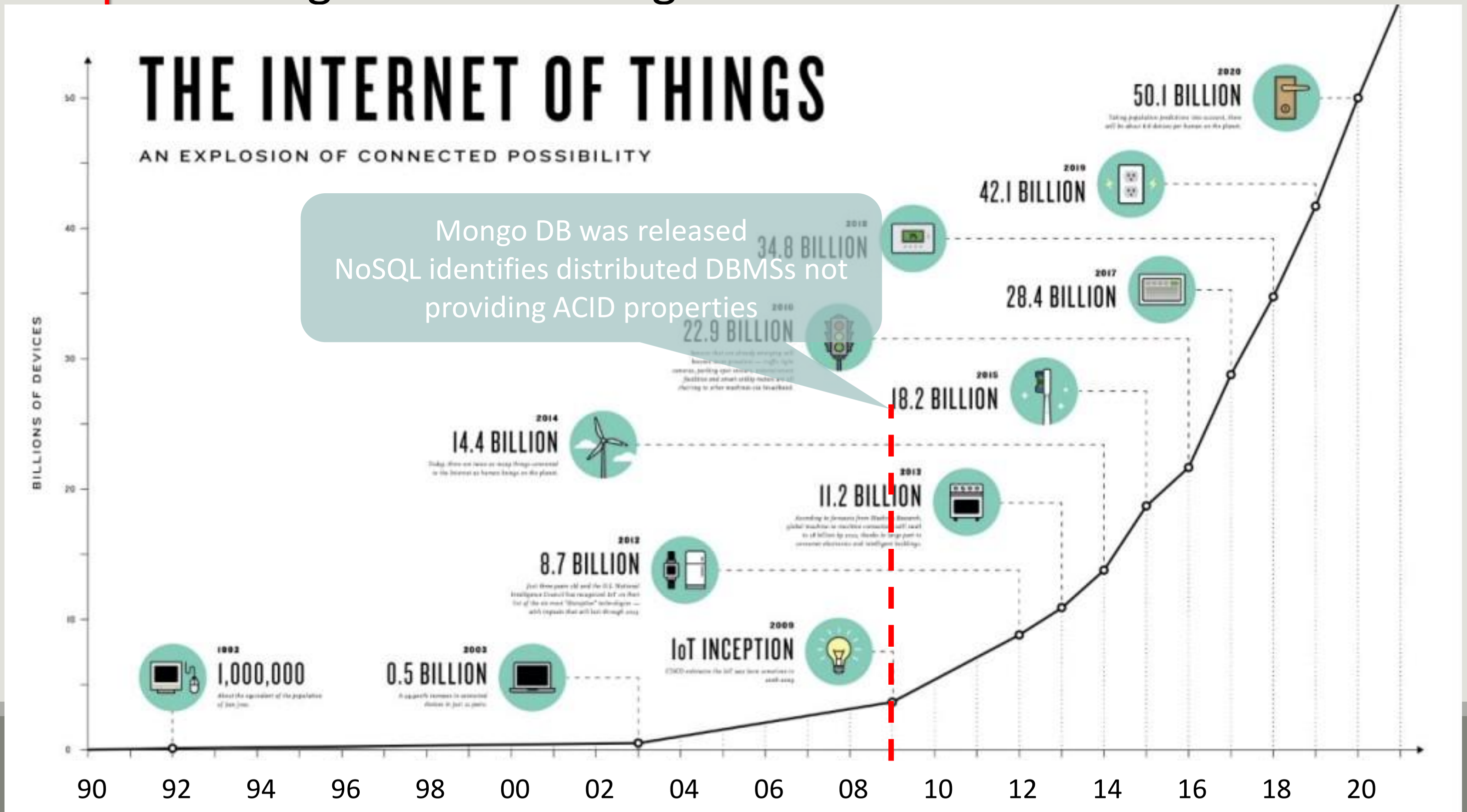
- Pressing user needs: users ask for quantitative information in the data-driven economy
- Data Availability: Internet of Things
- Computational power: Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

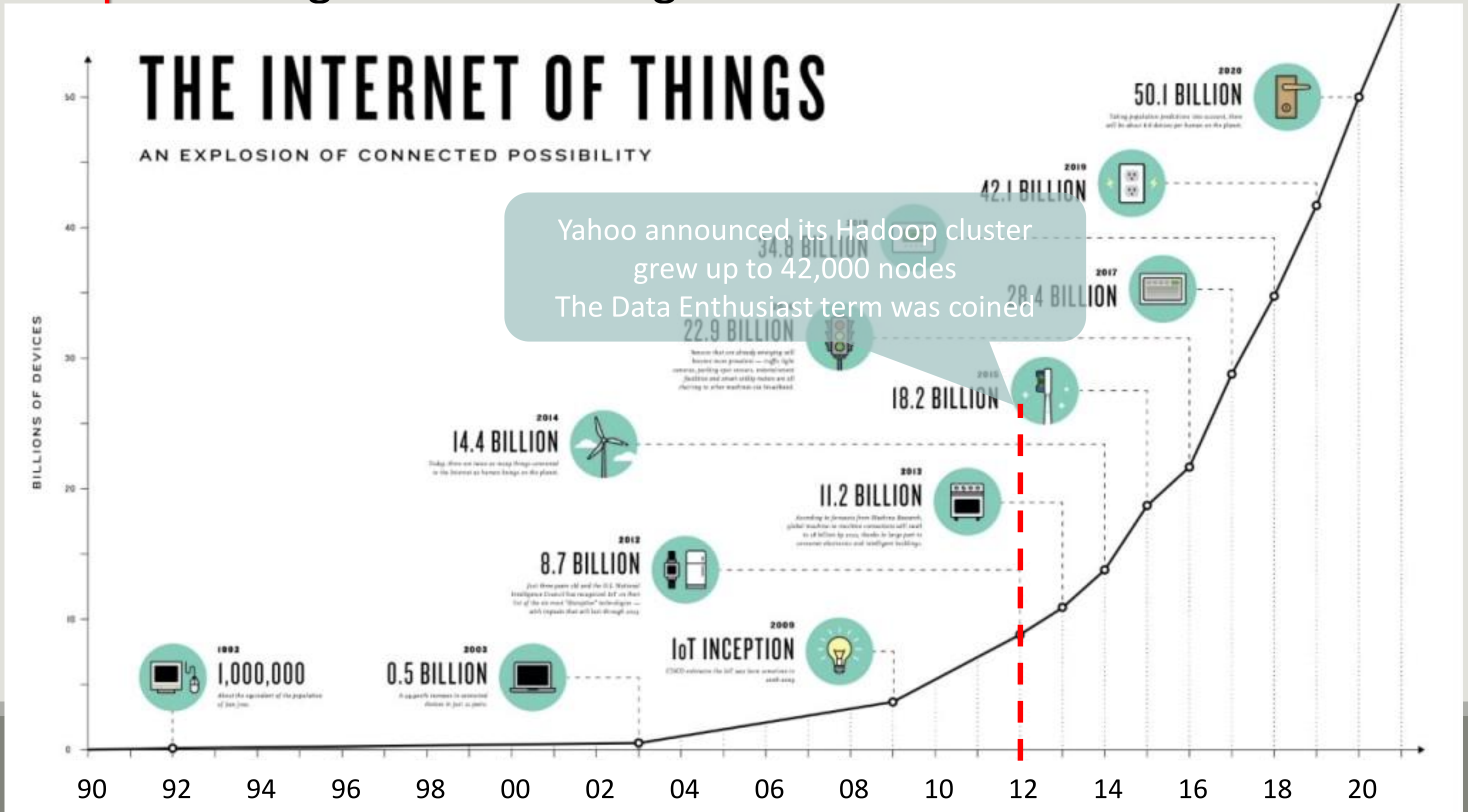
- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies



Where does Data science come from?

The enabler factors of this revolution are:

- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies

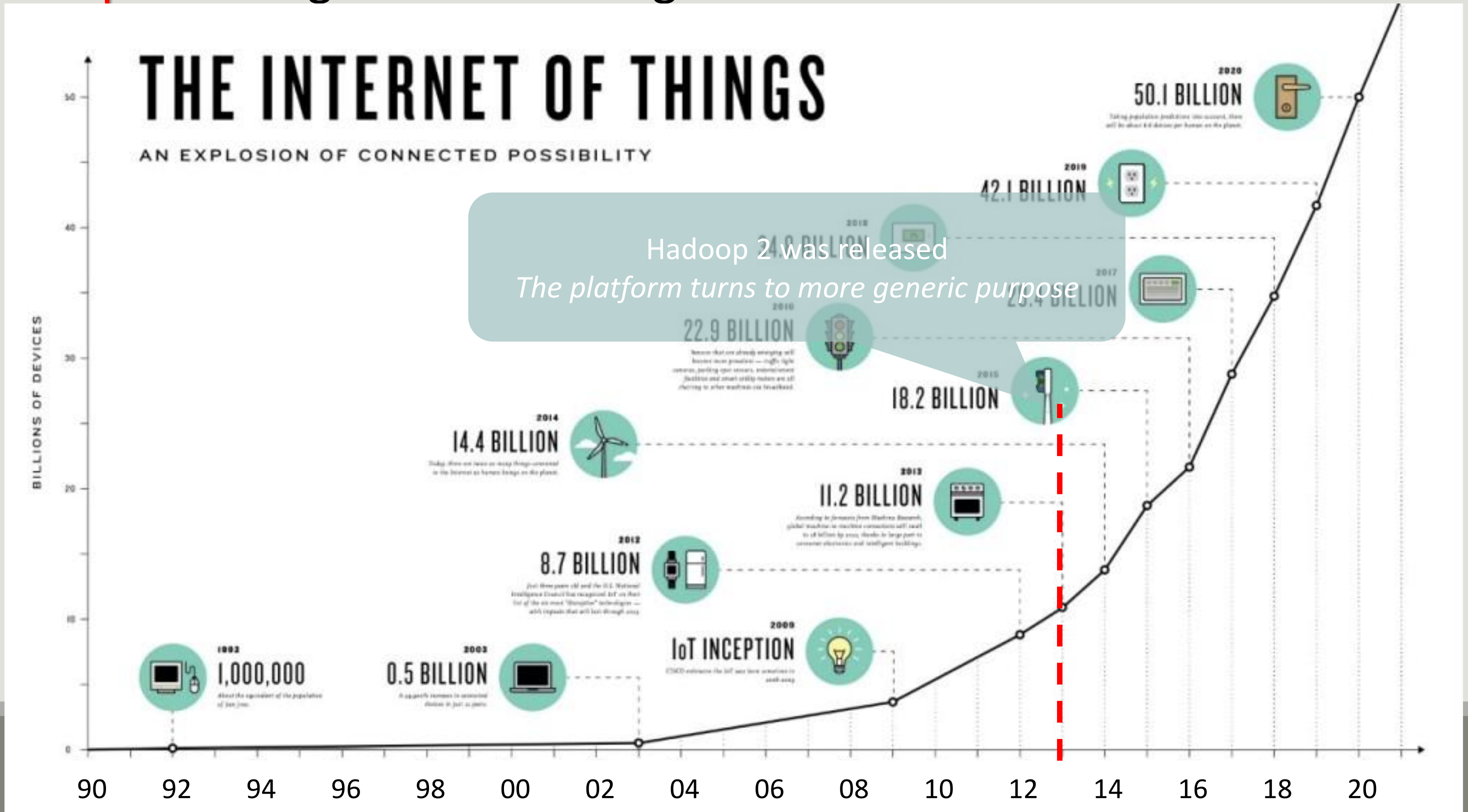


Yahoo announced its Hadoop cluster grew up to 42,000 nodes
The Data Enthusiast term was coined

Where does Data science come from?

The enabler factors of this revolution are:

- **Pressing user needs:** users ask for quantitative information in the data-driven economy
- **Data Availability:** Internet of Things
- **Computational power:** Big Data Technologies

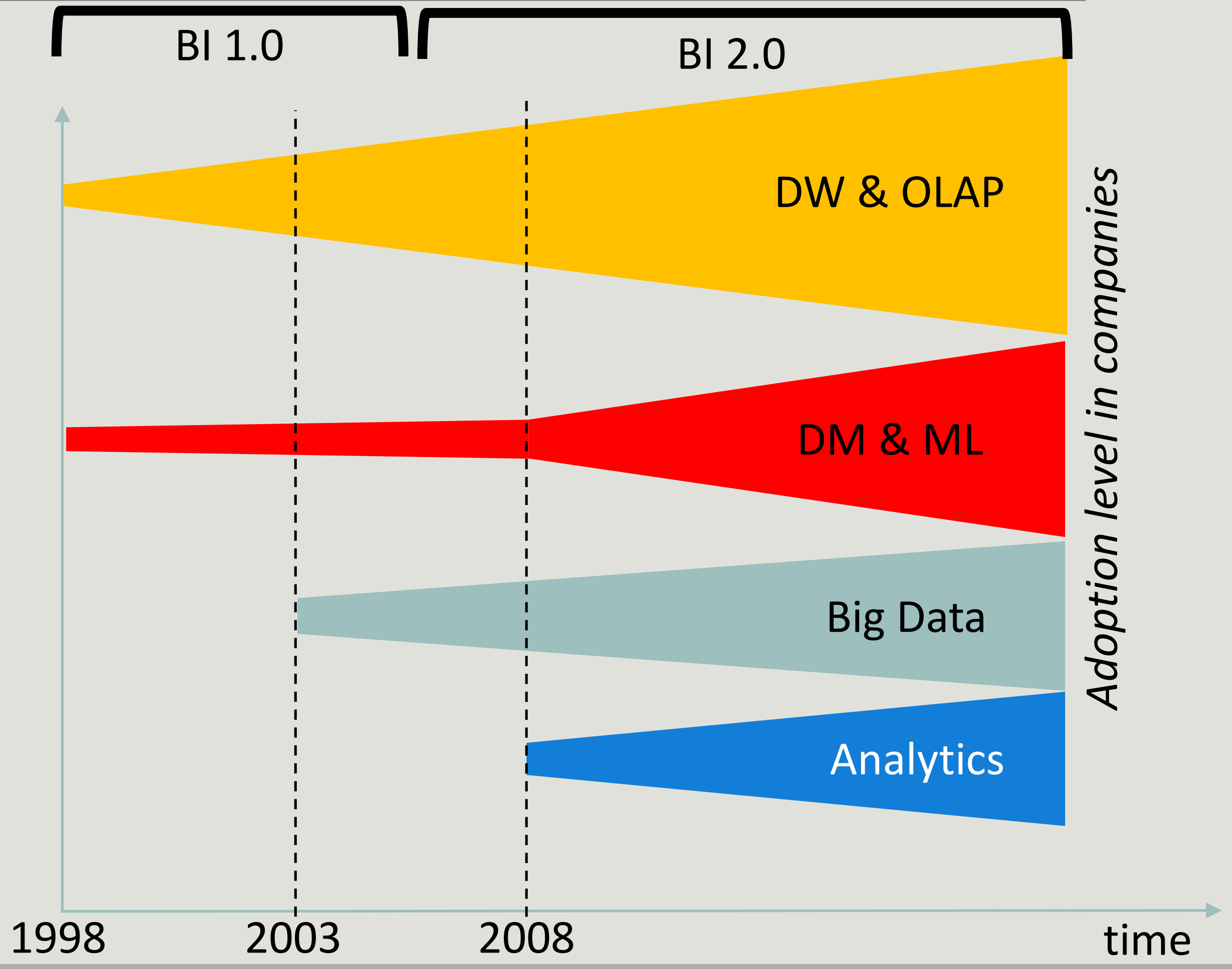


Big Data: the renaissance of Business Intelligence

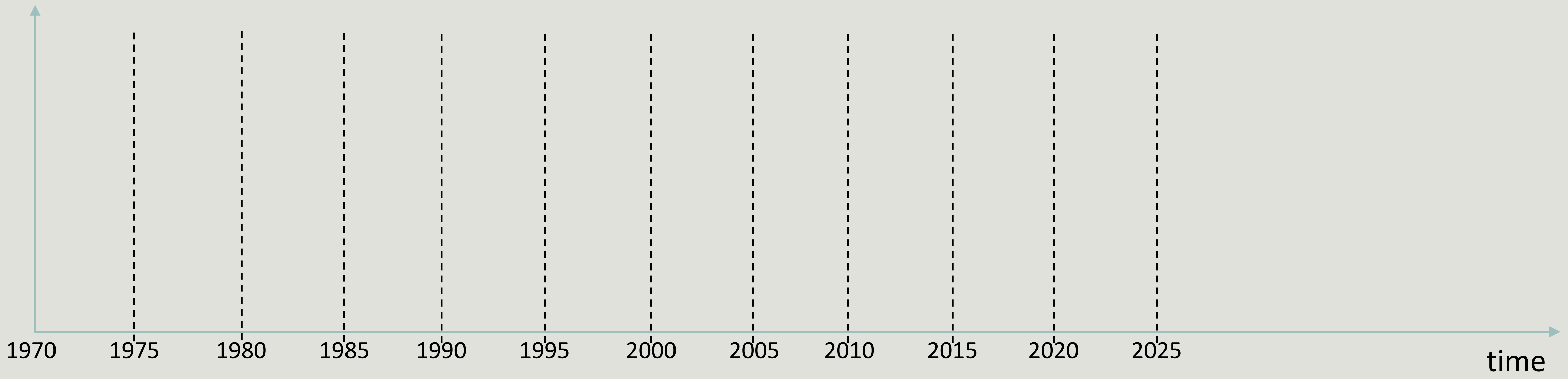
The main effect of Big Data raise is a renewed interest in the Data Analysis. While the technological gap can be bridged quickly, bridge the cultural gap for Data Analysis can take years (or even a new generation of managers).

- Most of the Big Data-related **technologies** are **NEW**
- Most of the Big Data-related **techniques** are already **KNOWN** and have been adapted to new technologies

A Time Line for Data Analysis



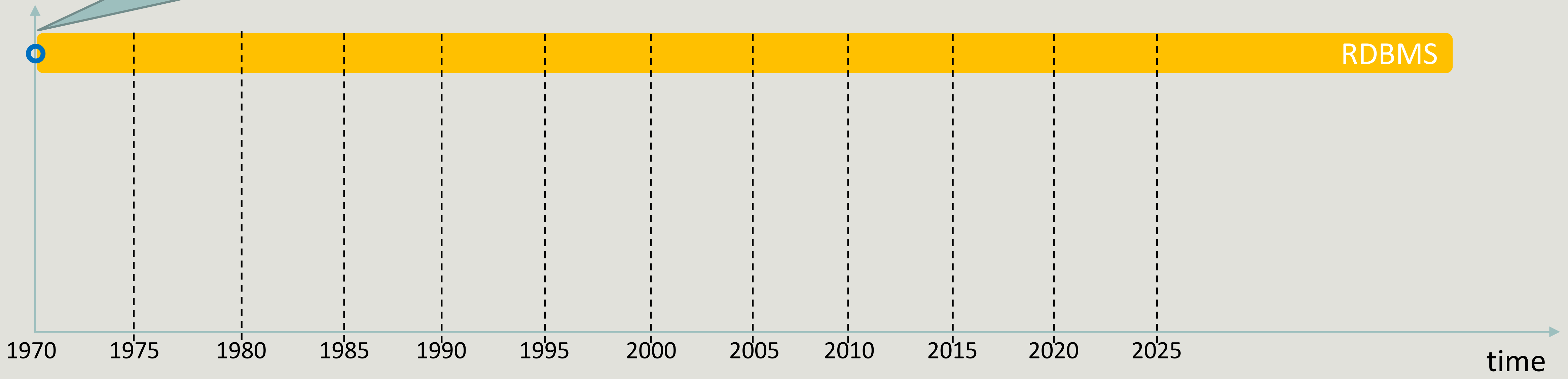
A Time Line for Data Analysis



A Time Line for Data Analysis

1970 E.F. Codd (IBM) publishes the relational model

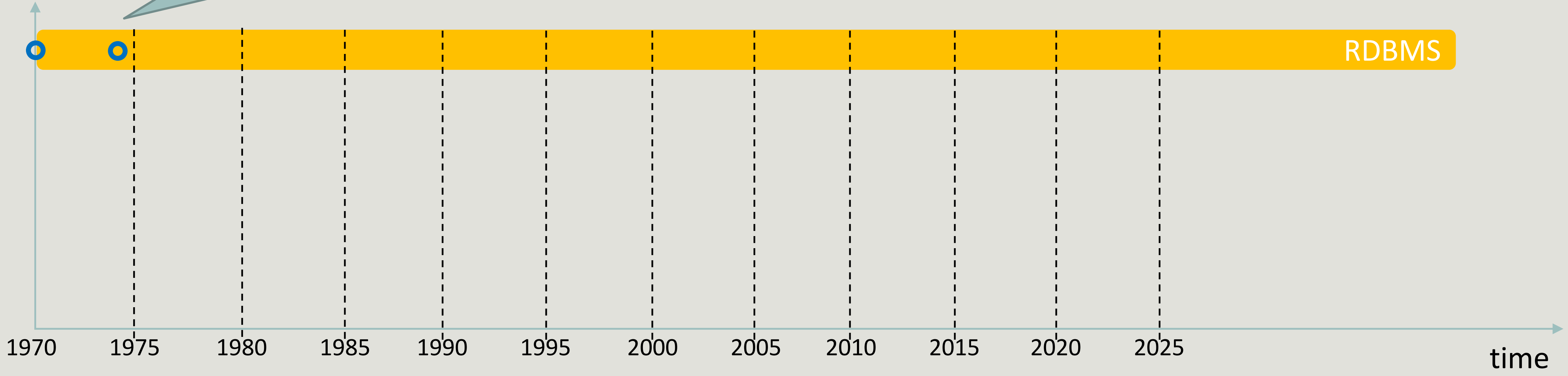
“It provides a means of describing data with its natural structure only--that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation on the other.”(Codd 1970)



A Time Line for Data Analysis

1974 SQL language was born - Donald Chamberlin (IBM)

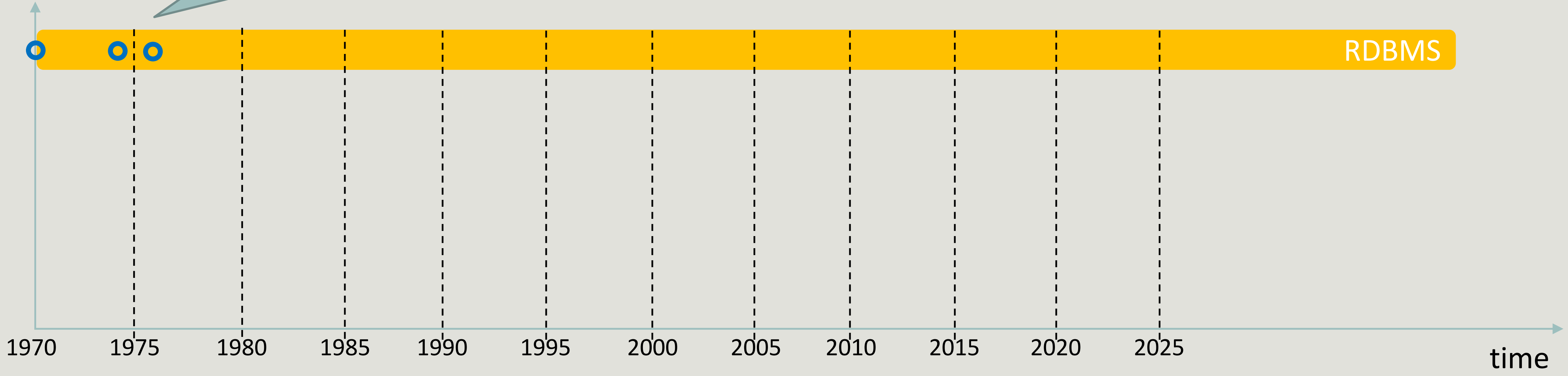
A standard language for querying relational DBMS



A Time Line for Data Analysis

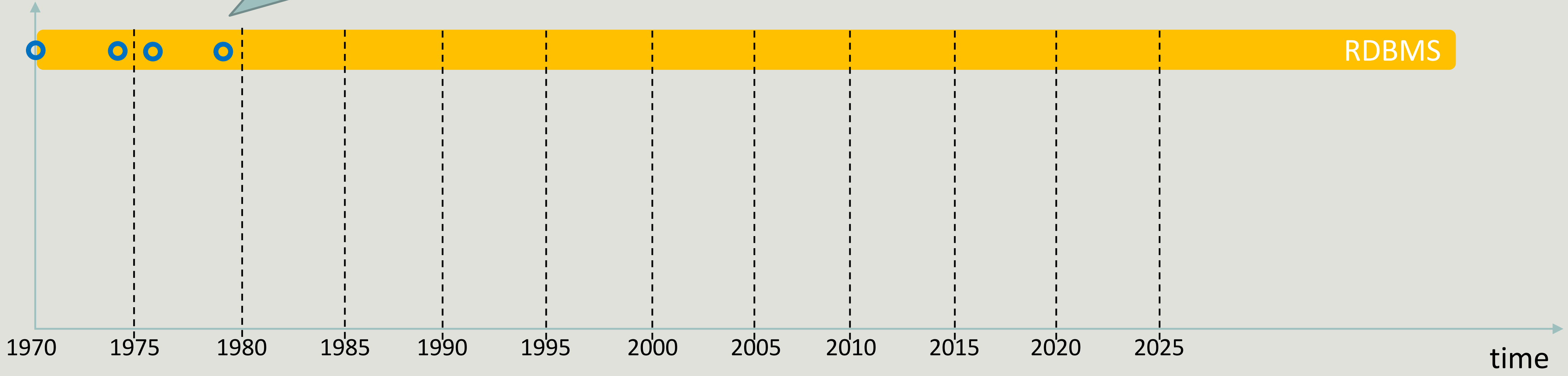
1976 Peter Chen publishes the Entity Relationship model

A conceptual model for relational databases



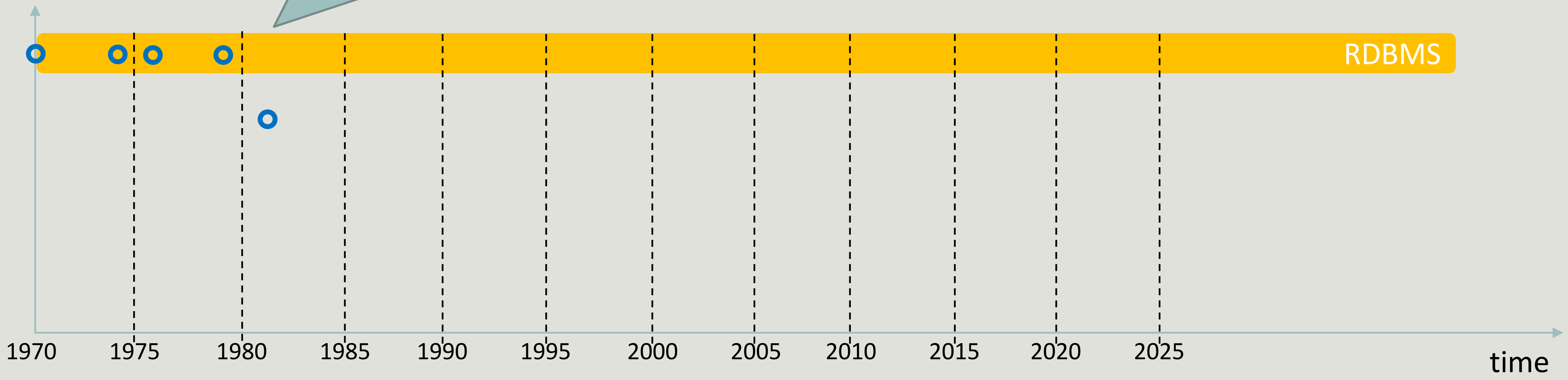
A Time Line for Data Analysis

1979 Larry Ellison launches the first ORACLE’s DBMS release



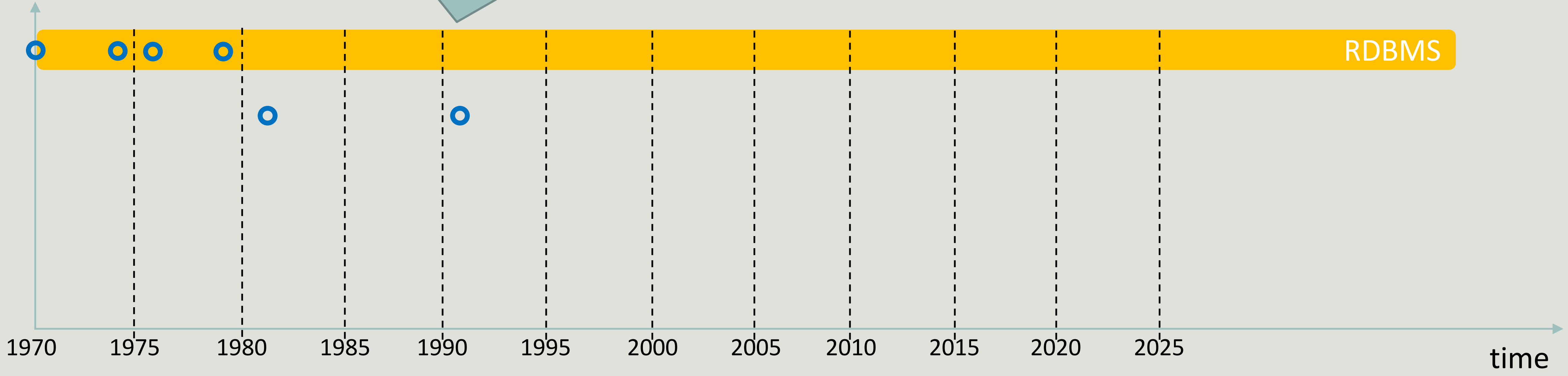
A Time Line for Data Analysis

1982 TCP/IP an the word "Internet" are defined



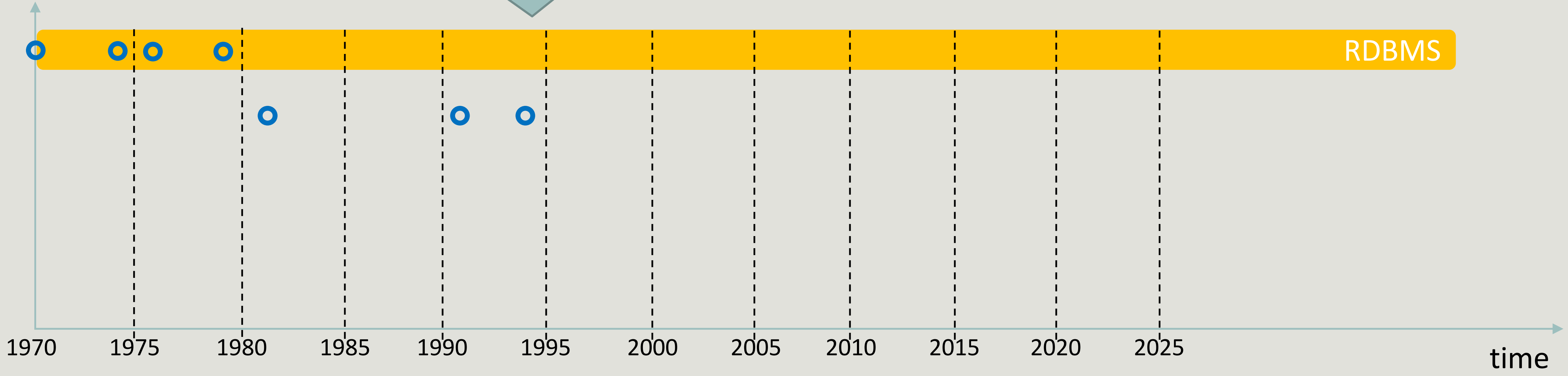
A Time Line for Data Analysis

1991 Il CERN announces internet



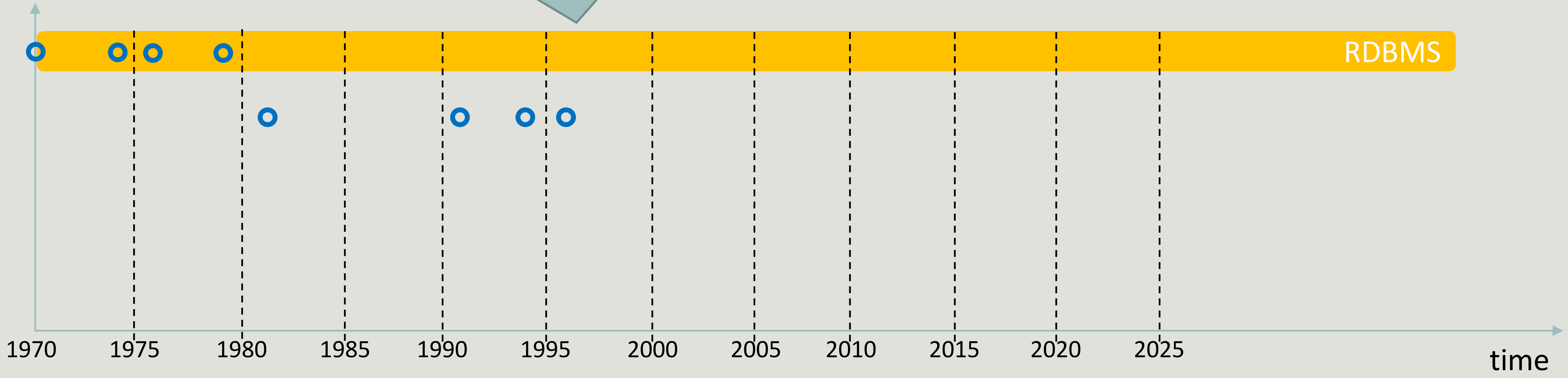
A Time Line for Data Analysis

1994 Jeff Bezos opens Amazon



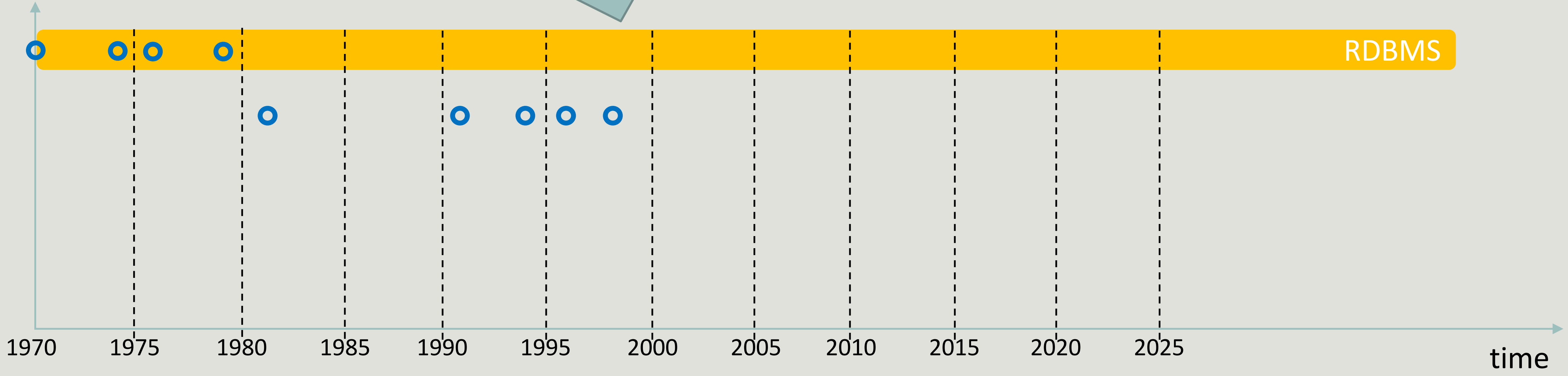
A Time Line for Data Analysis

1996 10 milions of computers are connected to the Internet



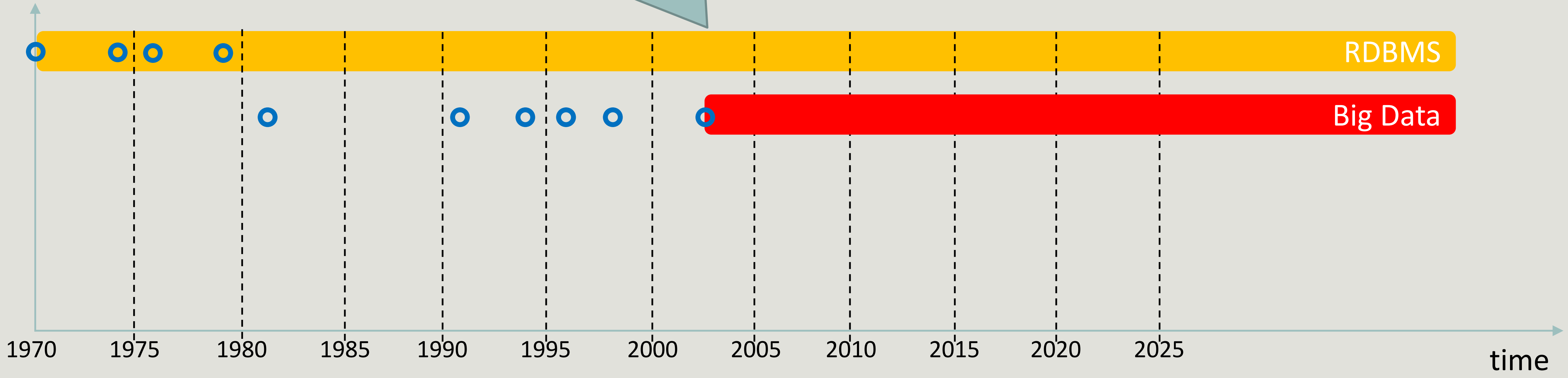
A Time Line for Data Analysis

1998 Larry Page and Sergey Brin open Google



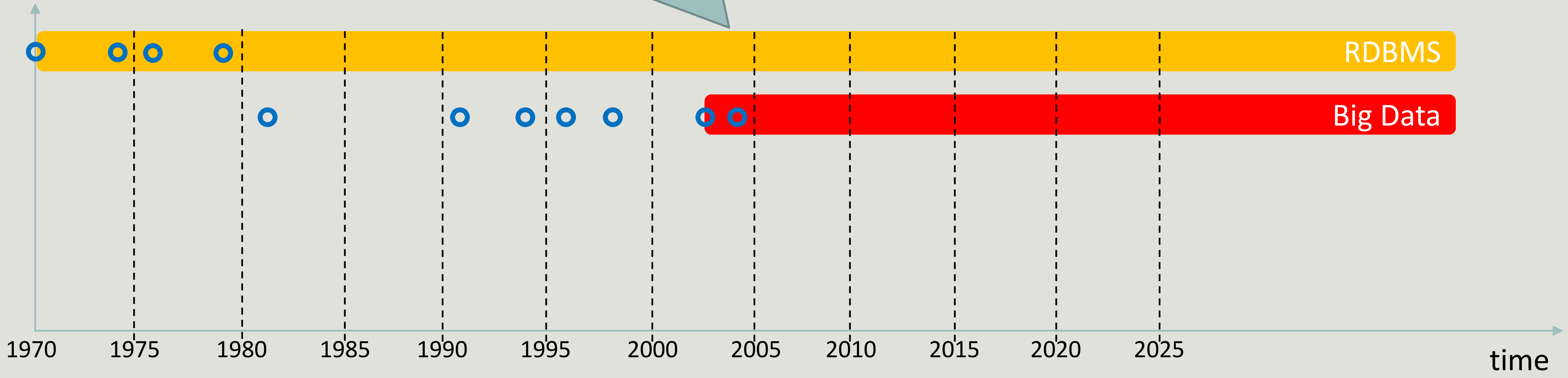
A Time Line for Data Analysis

2003 Google File System was born



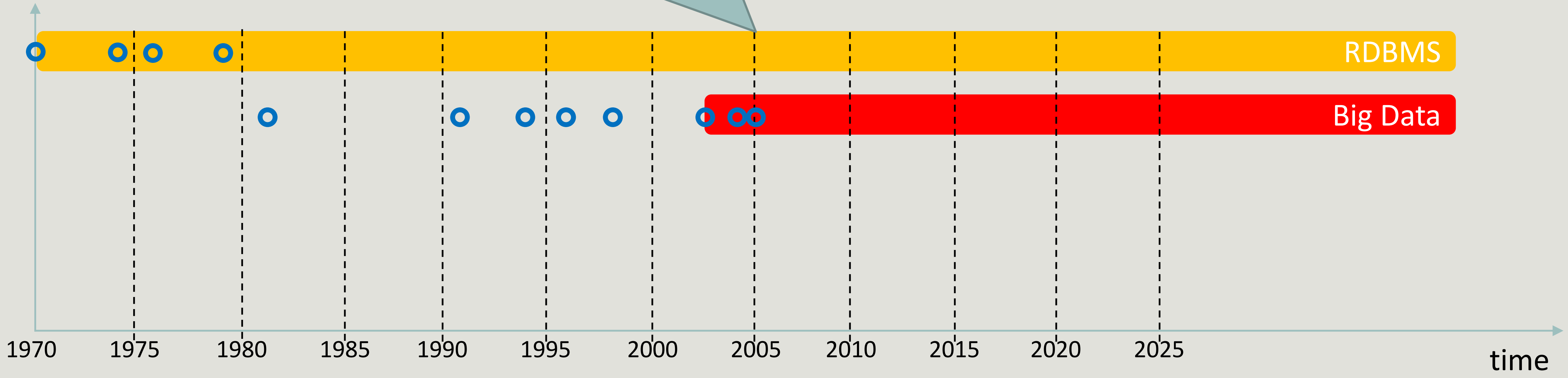
A Time Line for Data Analysis

2004 map Reduce (Google) was born
A distributed computation programming language



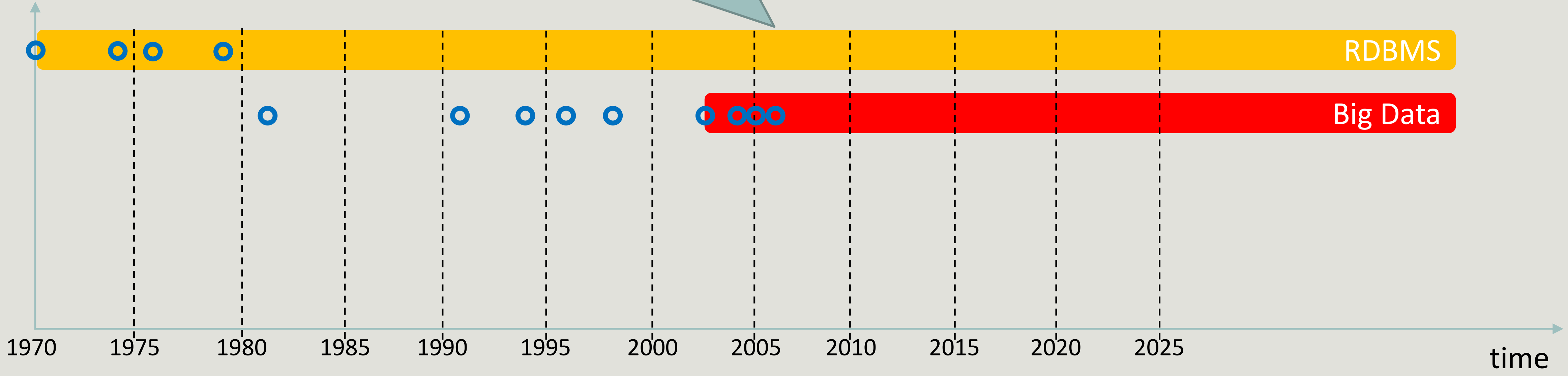
A Time Line for Data Analysis

2005 Hadoop 1 (Apache) was born
Based on Google papers, it is a Big Data open source platform for the web crawler Nutch



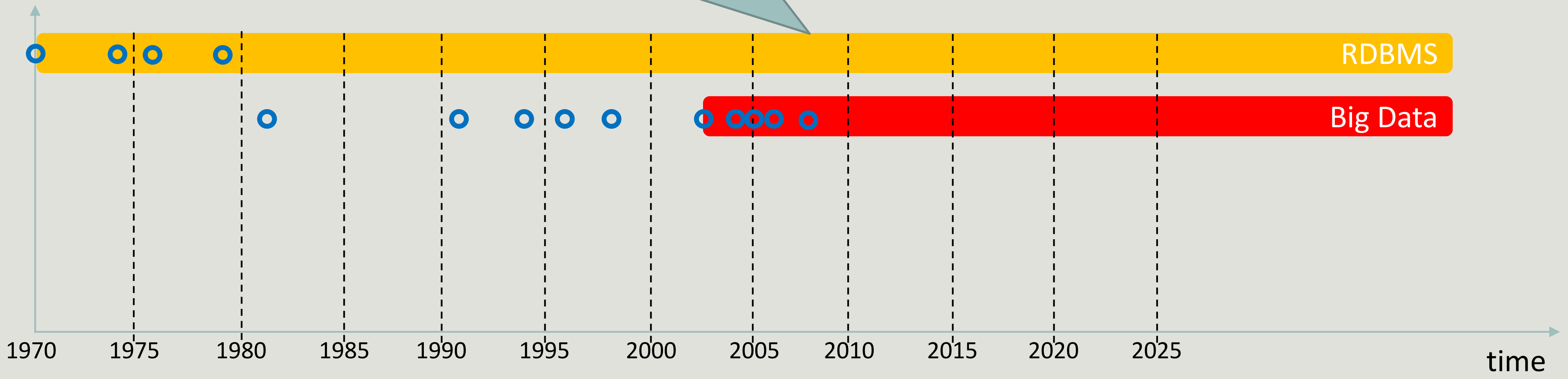
A Time Line for Data Analysis

2006 Amazon Web Services (AWS) was born
everyone is talking about cloud computing (conied on 1997)



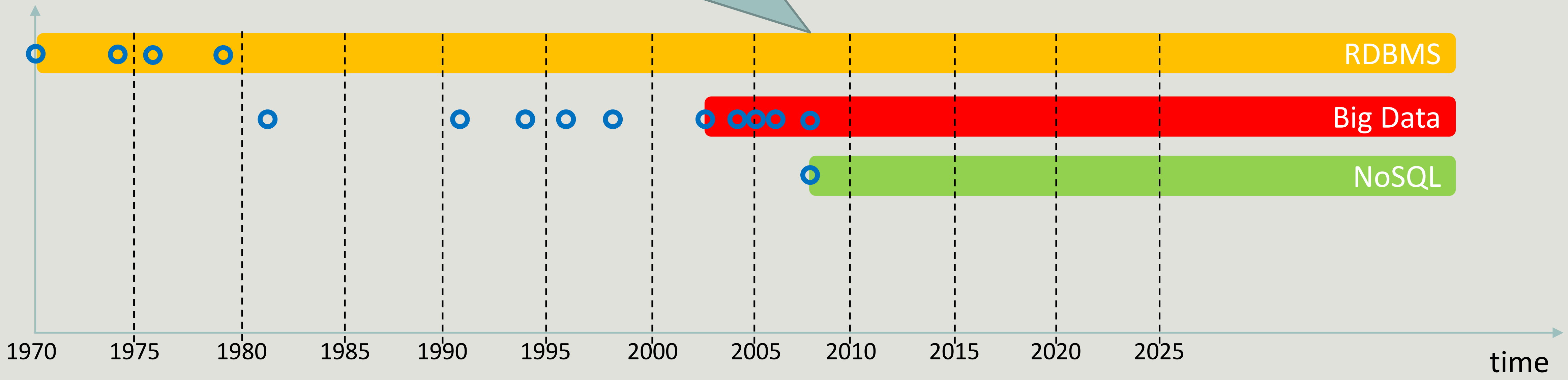
A Time Line for Data Analysis

2008 Data Science becomes trendy
DJ Patil states: "*Data Scientist is The Sexiest Job of the 21st Century*"



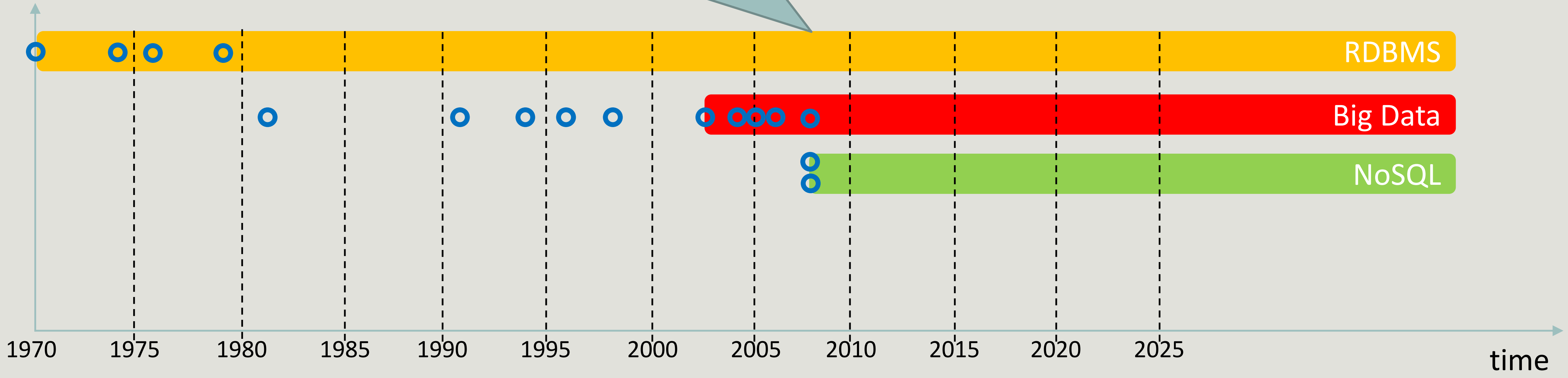
A Time Line for Data Analysis

2008 Cassandra (Facebook) was born
An open source non-relational DBMS (wide column)



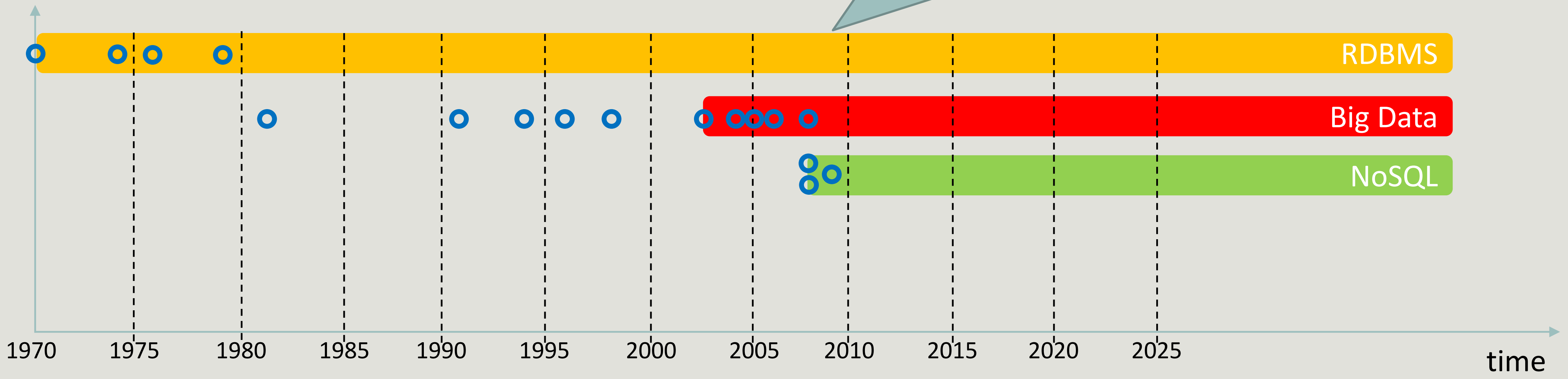
A Time Line for Data Analysis

2008 Voldemort (Linkedin) was born
An open source non-relational DBMS (key-value)



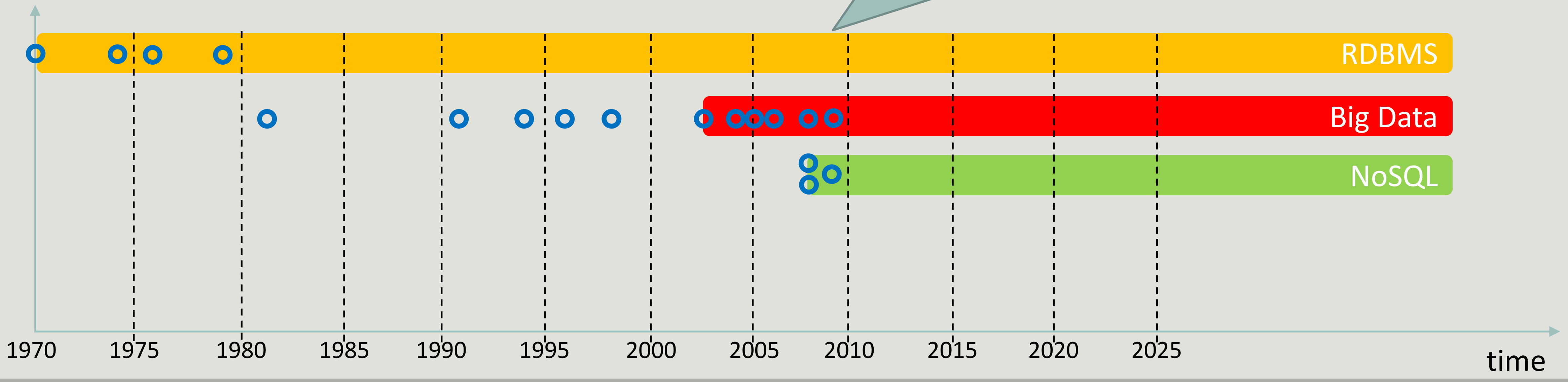
A Time Line for Data Analysis

2009 MongoDB was born
An open source non-relational DBMS (document-based)



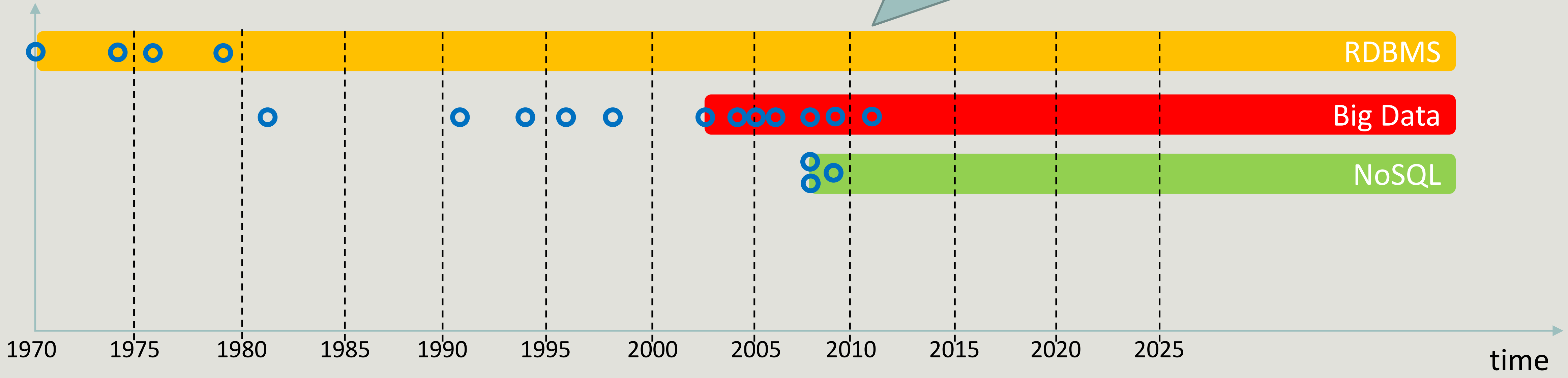
A Time Line for Data Analysis

2009 IoT was born
According to CISCO more things than persons are connected on Internet



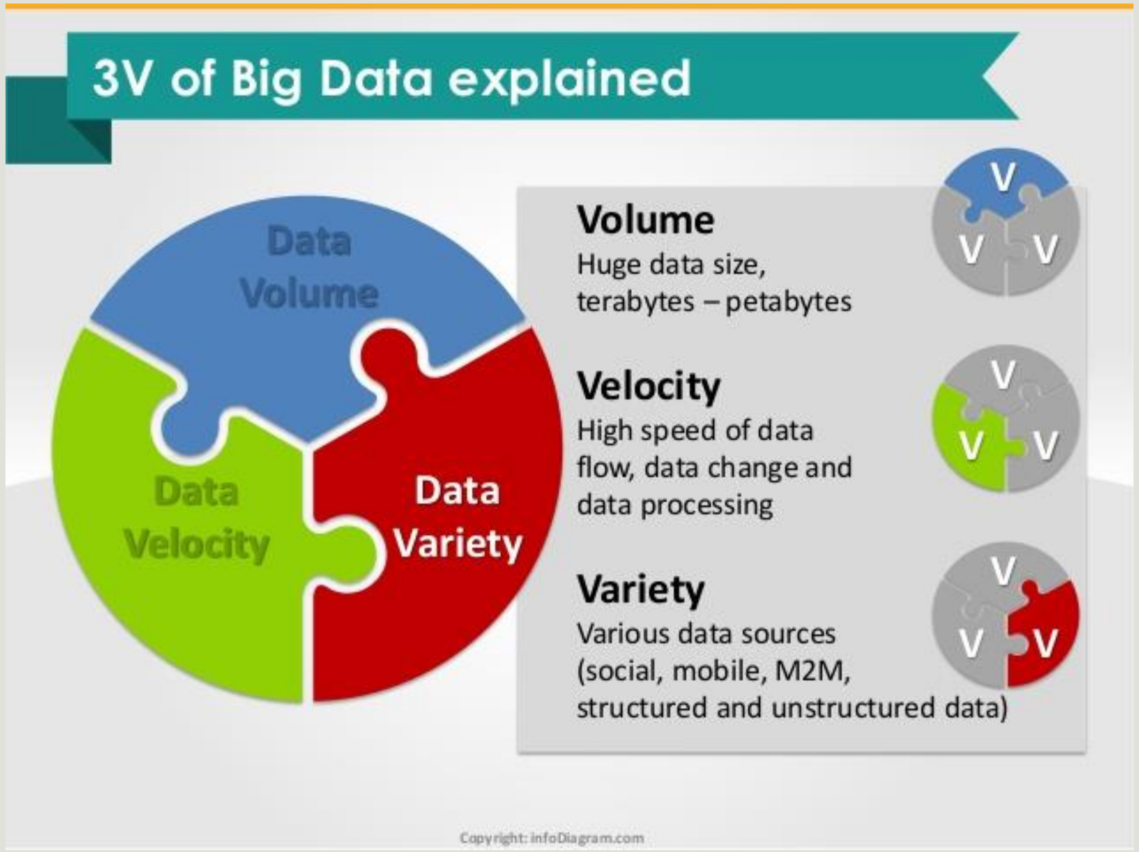
A Time Line for Data Analysis

2011 HIVE was born
A relational DBMS on a Big Data platform



Big Data: a definition

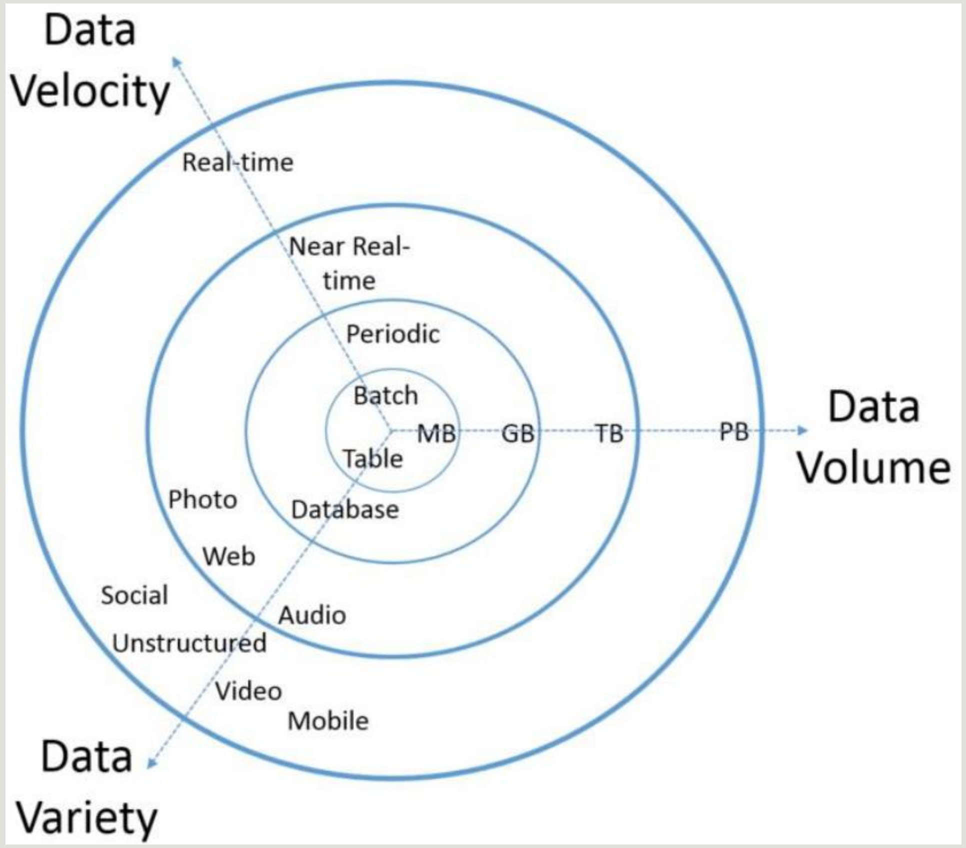
Big Data are dataset with the following features, ***not necessarily all!***



Volume
Huge data size, terabytes – petabytes

Velocity
High speed of data flow, data change and data processing

Variety
Various data sources (social, mobile, M2M, structured and unstructured data)



Big Data: volume

Volume: Terabyte or Petabyte so that they exceed the the processing capacity of conventional database systems.

- Some examples:
  - Walmart: 1 million transaction per hour (2010)
  - eBay: data throughput reaches 100 petabytes per day (2013)
  - Facebook: 40 billion photos (2010); 250PB data warehouse with 600TB added to the warehouse every day (2013)
  - 500 millions of tweet per day (in 2013)

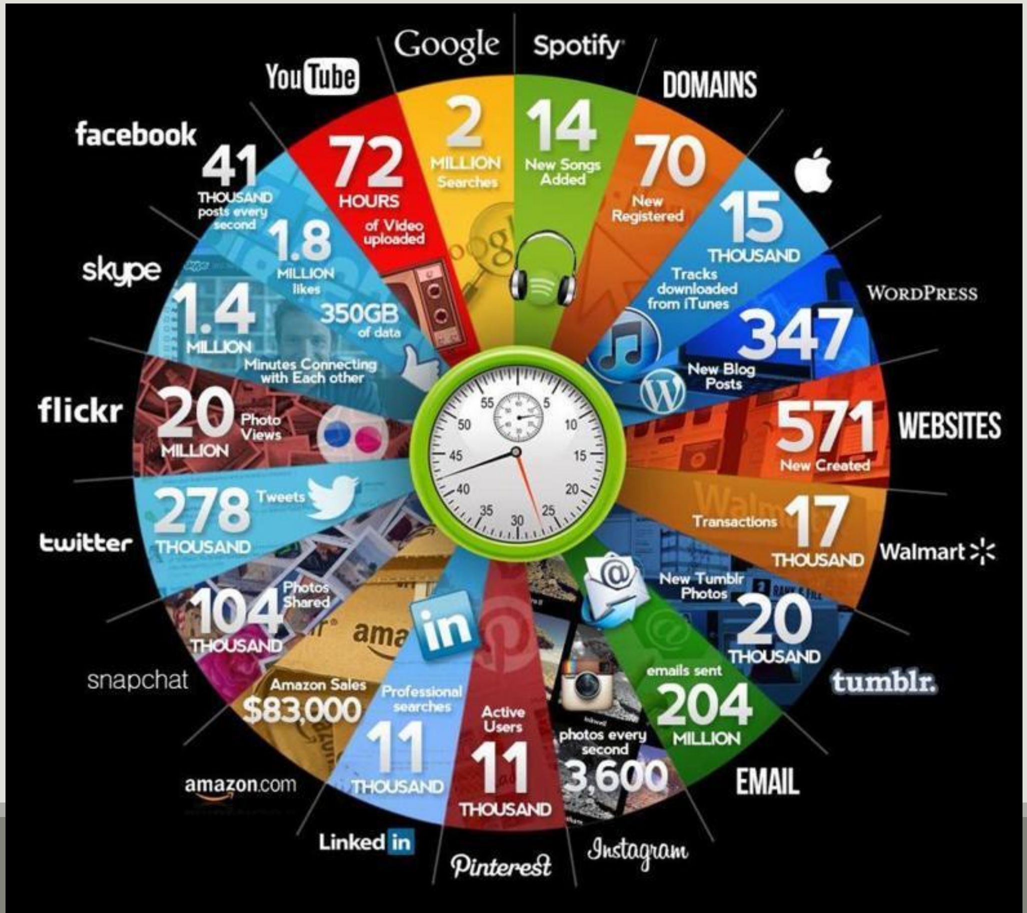
The hard disk of a desktop PC can store up to 1 TB

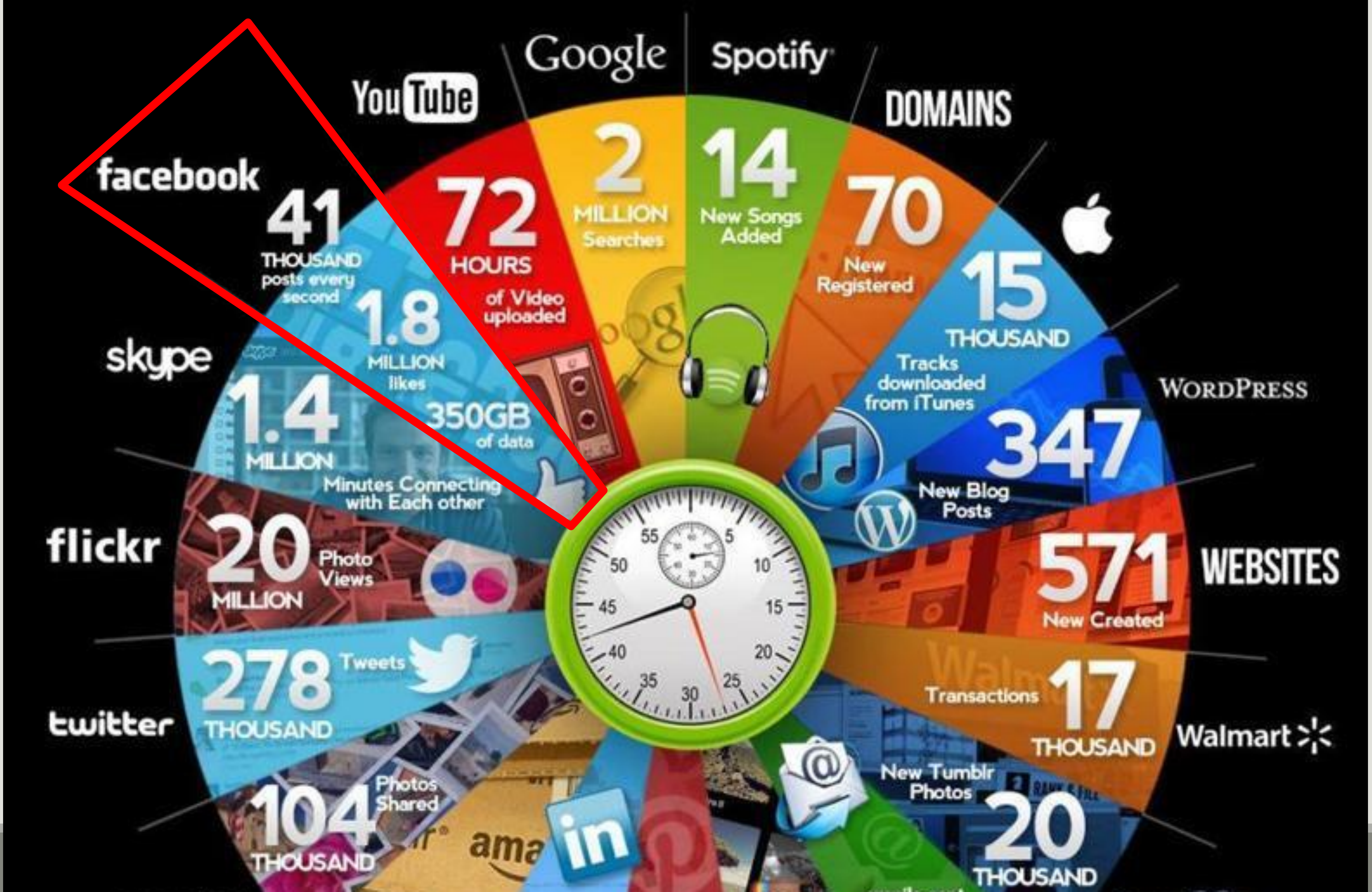
terabyte	TB	10^{12}	1 HD
petabyte	PB	10^{15}	1000 HD
exabyte	EB	10^{18}	1 milion of HD
zettabyte	ZB	10^{21}	1 bilion of HD

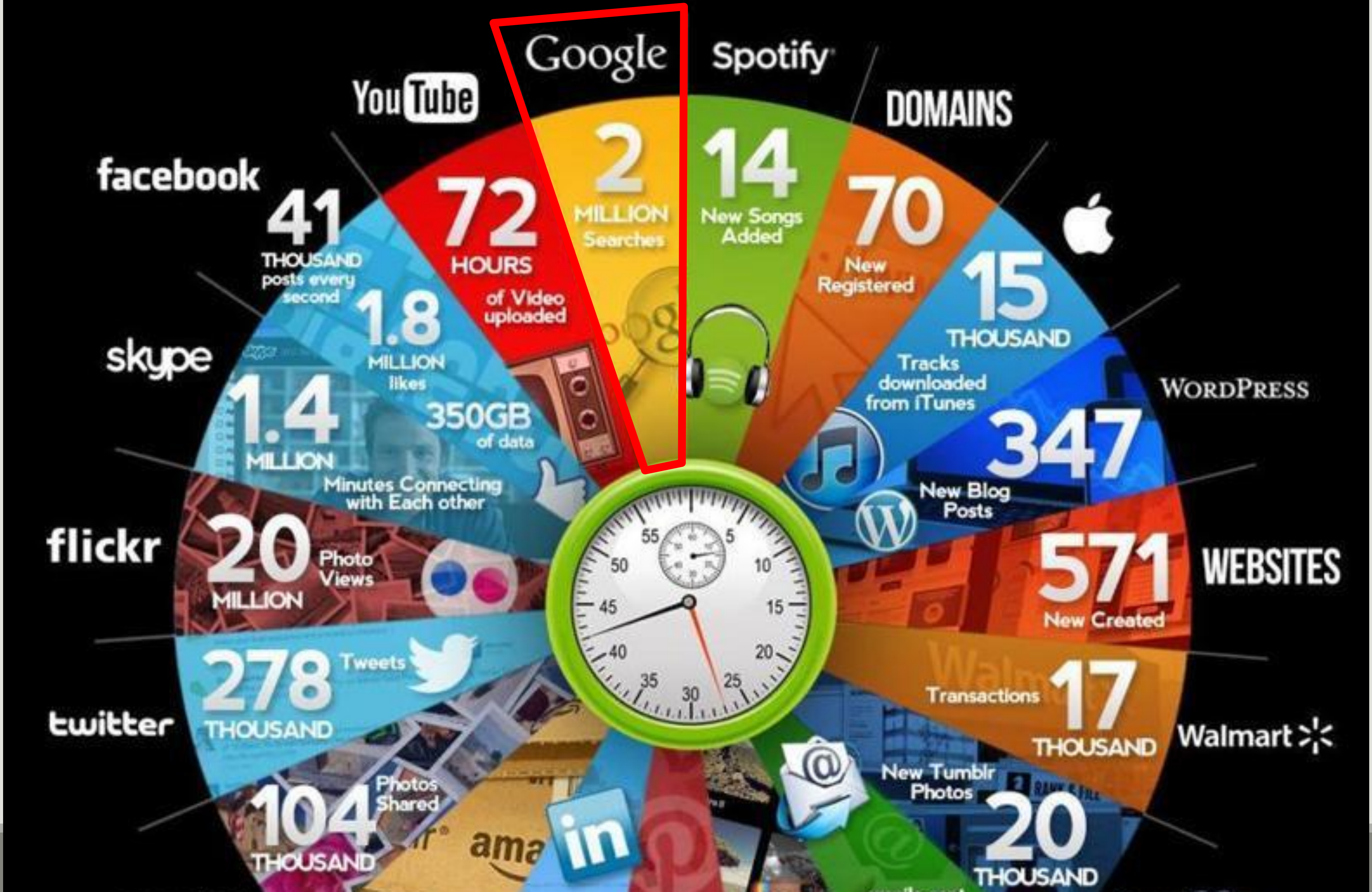
Big Data: velocity

Velocity: mobile and personal devices, IoT digital transactions produce data at a rate higher than traditional information systems

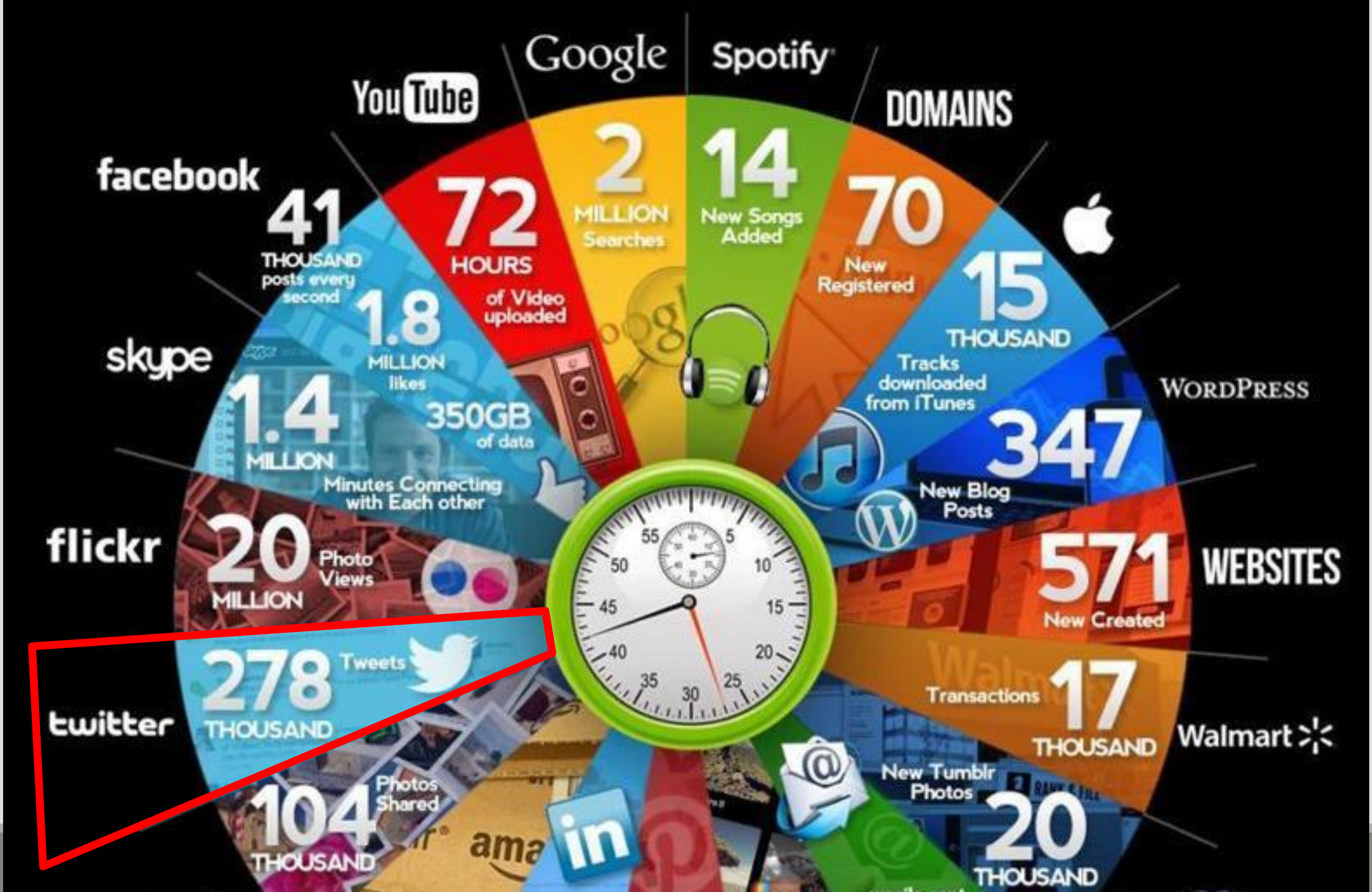
What does it happen in internet in 60 seconds?




facebook
41 THOUSAND posts every second
1.8 MILLION likes
350GB of data
YouTube
72 HOURS of Video uploaded
Google
2 MILLION Searches
Spotify
14 New Songs Added
DOMAINS
70 New Registered
15 THOUSAND Tracks downloaded from iTunes
WordPress
347 New Blog Posts
WEBSITES
571 New Created
Walmart
17 THOUSAND Transactions
20 THOUSAND New Tumblr Photos
skype
1.4 MILLION Minutes Connecting with Each other
flickr
20 MILLION Photo Views
twitter
278 THOUSAND Tweets
104 THOUSAND Photos Shared
55
5
10
15
20
25
30
35
40
45
50



Google
2 MILLION Searches
Spotify
14 New Songs Added
DOMAINS
70 New Registered
15 THOUSAND Tracks downloaded from iTunes
WORDPRESS
347 New Blog Posts
571 New Created
WEBSITES
Transactions 17 THOUSAND
Walmart
New Tumblr Photos
20 THOUSAND
YouTube
72 HOURS of Video uploaded
facebook
41 THOUSAND posts every second
1.8 MILLION likes
350GB of data
skype
1.4 MILLION Minutes Connecting with Each other
flickr
20 MILLION Photo Views
278 THOUSAND Tweets
twitter
104 THOUSAND Photos Shared
55
5
50
10
45
15
40
20
35
25
30


Google
2 MILLION Searches
Spotify
14 New Songs Added
DOMAINS
70 New Registered
15 THOUSAND Tracks downloaded from iTunes
WORDPRESS
347 New Blog Posts
WEBSITES
571 New Created
Walmart
Transactions 17 THOUSAND
New Tumblr Photos 20 THOUSAND
YouTube
72 HOURS of Video uploaded
facebook
41 THOUSAND posts every second
1.8 MILLION likes
350GB of data
skype
1.4 MILLION Minutes Connecting with Each other
flickr
20 MILLION Photo Views
twitter
278 THOUSAND Tweets
104 THOUSAND Photos Shared
55
5
10
15
20
25
30
35
40
45
50

Big Data: Variety

- Variety: Data is extremely heterogeneous, in the format in which are represented, but also and in the way they represent information, both at the intensional and extensional level
  - E.g., text from social networks, sensor data, logs from web applications, databases, XML documents, RDF data, etc.
  - Data format ranges therefore from structured (e.g, relational databases) to semistructured (e.g., XML documents), to unstructured (e.g., text documents)

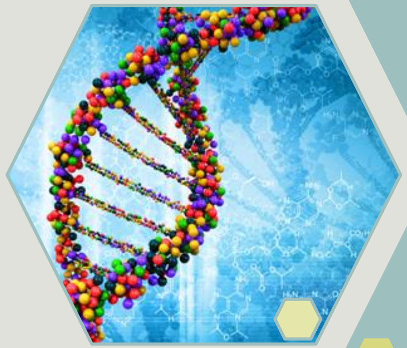
- *Veracity*: the quality level of data is very heterogeneous and in many cases a well-defined schema is not provided
  - The **schema less** concept emphasizes such heterogeneity

Big Data… Many Definitions

The Multiple V's: Data that brings challenges in Volume (size), Velocity(speed), Variety(formats), Veracity(accuracy), as well as Visualization, Value, Vendors, etc.
McKinsey: "Datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze."!
Economist: "Society has more information than ever before and we can do things when we have a large body of information that simply we could not do when we only have smaller amounts"
Wikipedia: "Big data is a term for data sets that are so large or complex that traditional data processing application software is inadequate to deal with them. Big data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating and information privacy."
Adam Jacobs, **1010 data**: "Data whose size forces us to look beyond the tried-and-true methods that are prevalent"
Dan Law, **Altamira**: "Data of any type with potential value that exceeds the analytic capabilities of traditional stand-alone solutions", "Any data collection that cannot be managed as a single instance."

Who generates big data?

Web & Social



IoT

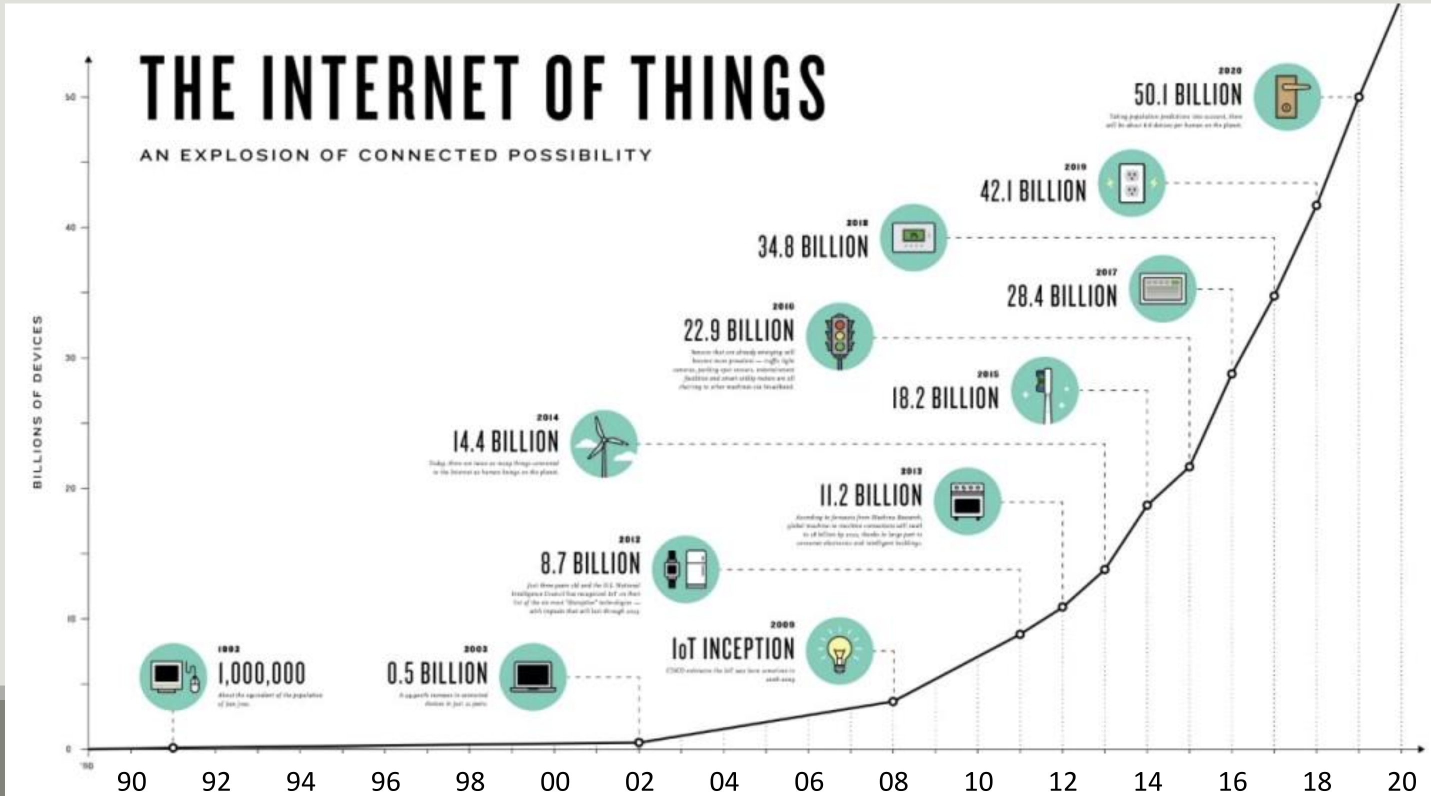
Data intensive analysis

genetics
meteo

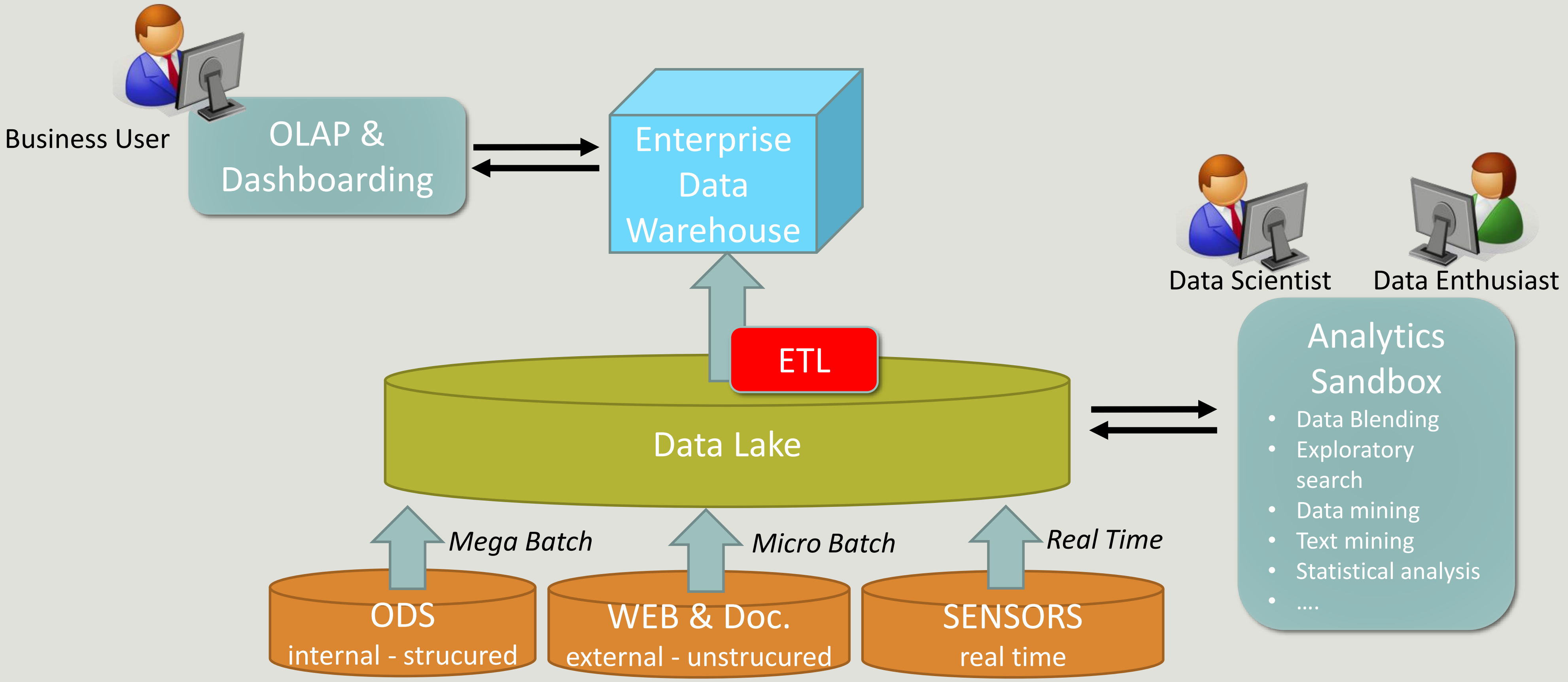
Who generates big data?

IoT – Internet of Things it is the evolution of the use of the Internet in which objects become recognizable and acquire intelligence because they can communicate data about themselves and access aggregate information from other

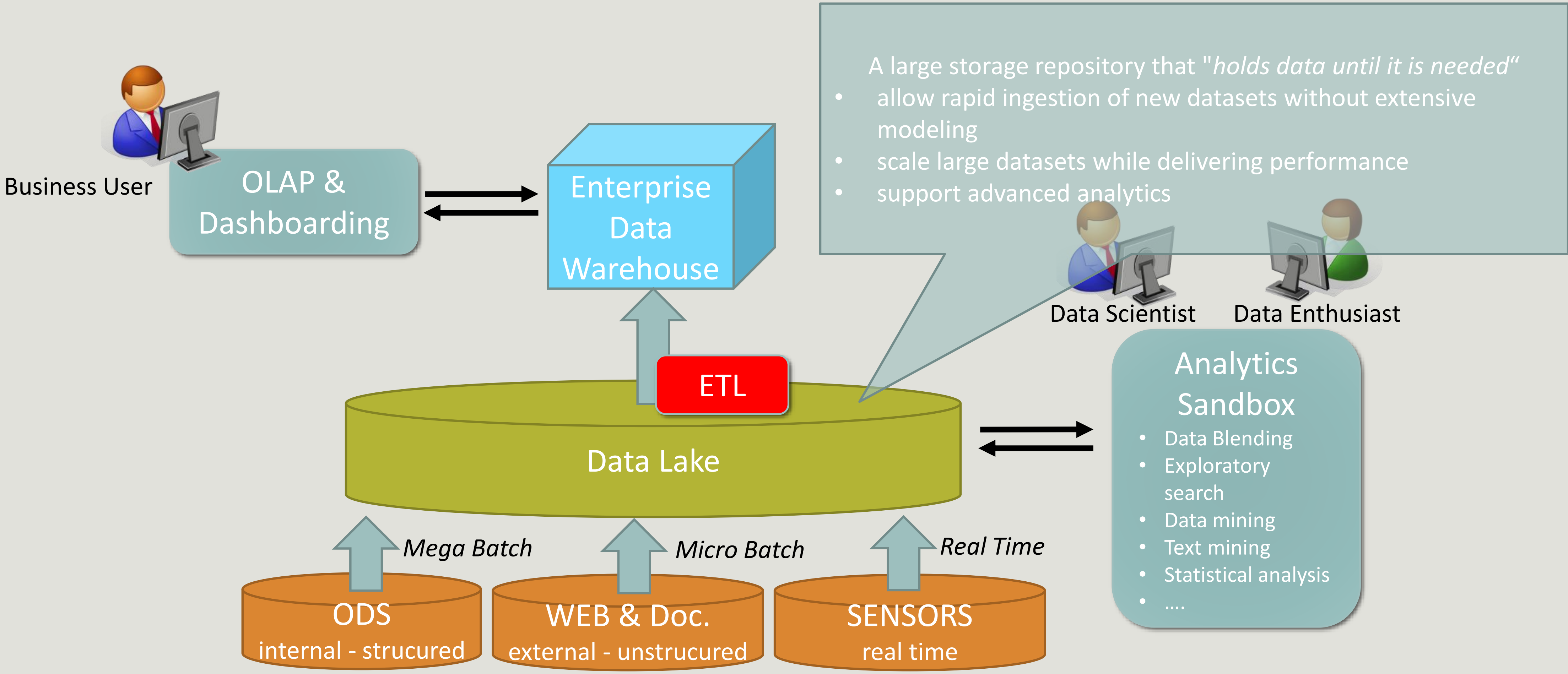
- The alarm sounds earlier in case of traffic
- The frozen food in the supermarket freezer signals that it is about to expire
- The industrial plant reports a high probability of failure within the next few hours



A data science infrastructure



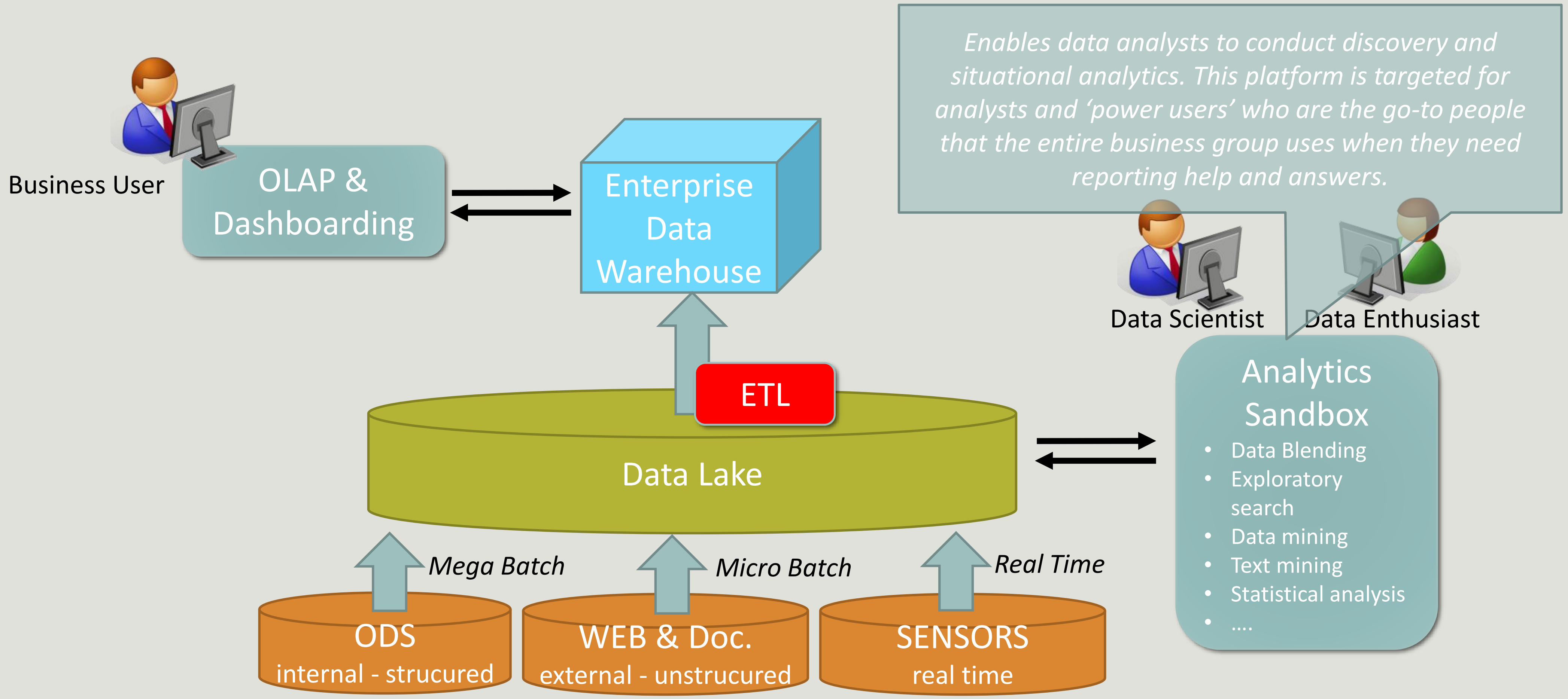
A data science infrastructure



A large storage repository that "*holds data until it is needed*"

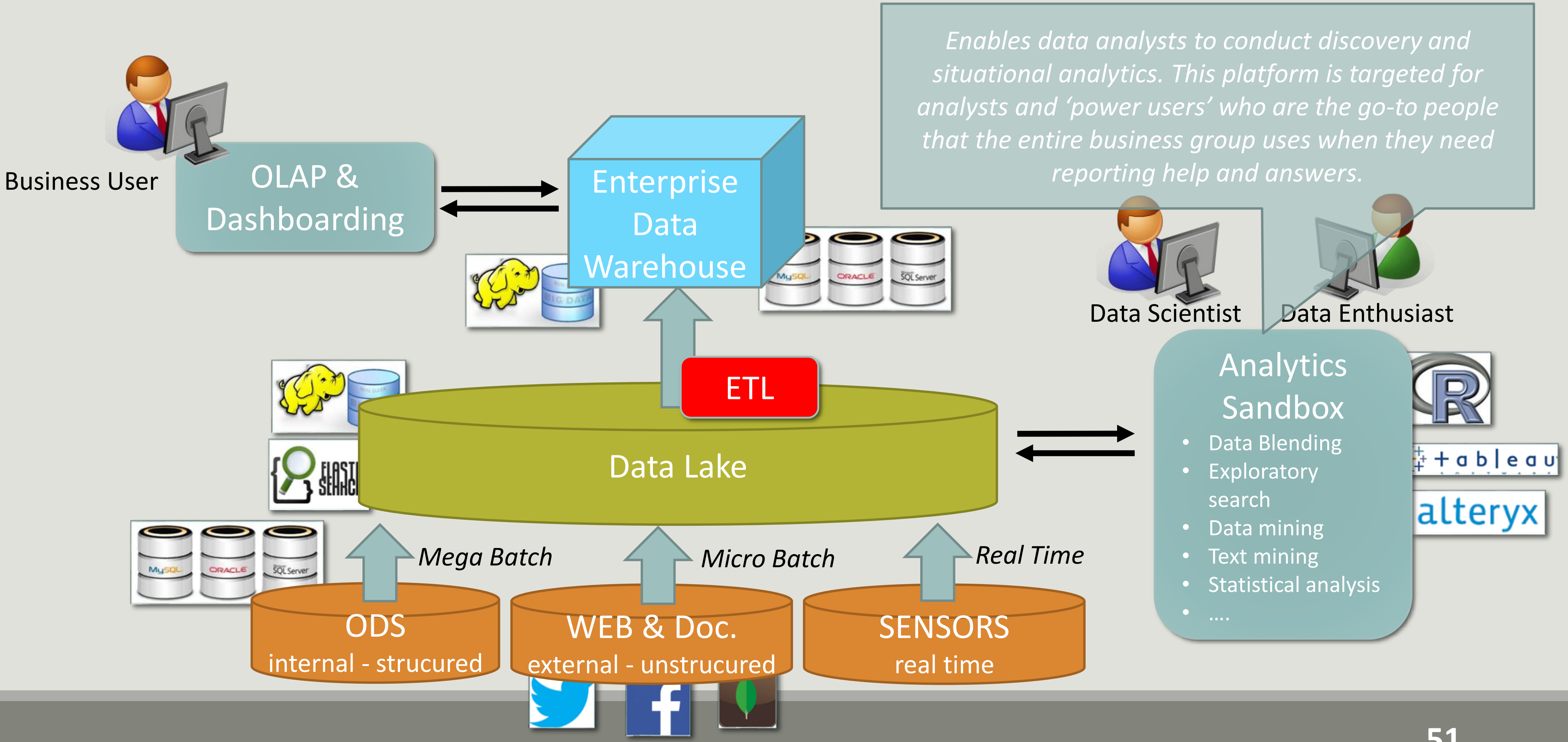
- allow rapid ingestion of new datasets without extensive modeling
- scale large datasets while delivering performance
- support advanced analytics

A data science infrastructure



Enables data analysts to conduct discovery and situational analytics. This platform is targeted for analysts and 'power users' who are the go-to people that the entire business group uses when they need reporting help and answers.
Business User
OLAP & Dashboarding
Enterprise Data Warehouse
Data Scientist
Data Enthusiast
ETL
Data Lake
Analytics Sandbox
• Data Blending
• Exploratory search
• Data mining
• Text mining
• Statistical analysis
• ….
Mega Batch
Micro Batch
Real Time
ODS internal - strucured
WEB & Doc. external - unstrucured
SENSORS real time

A data science infrastructure



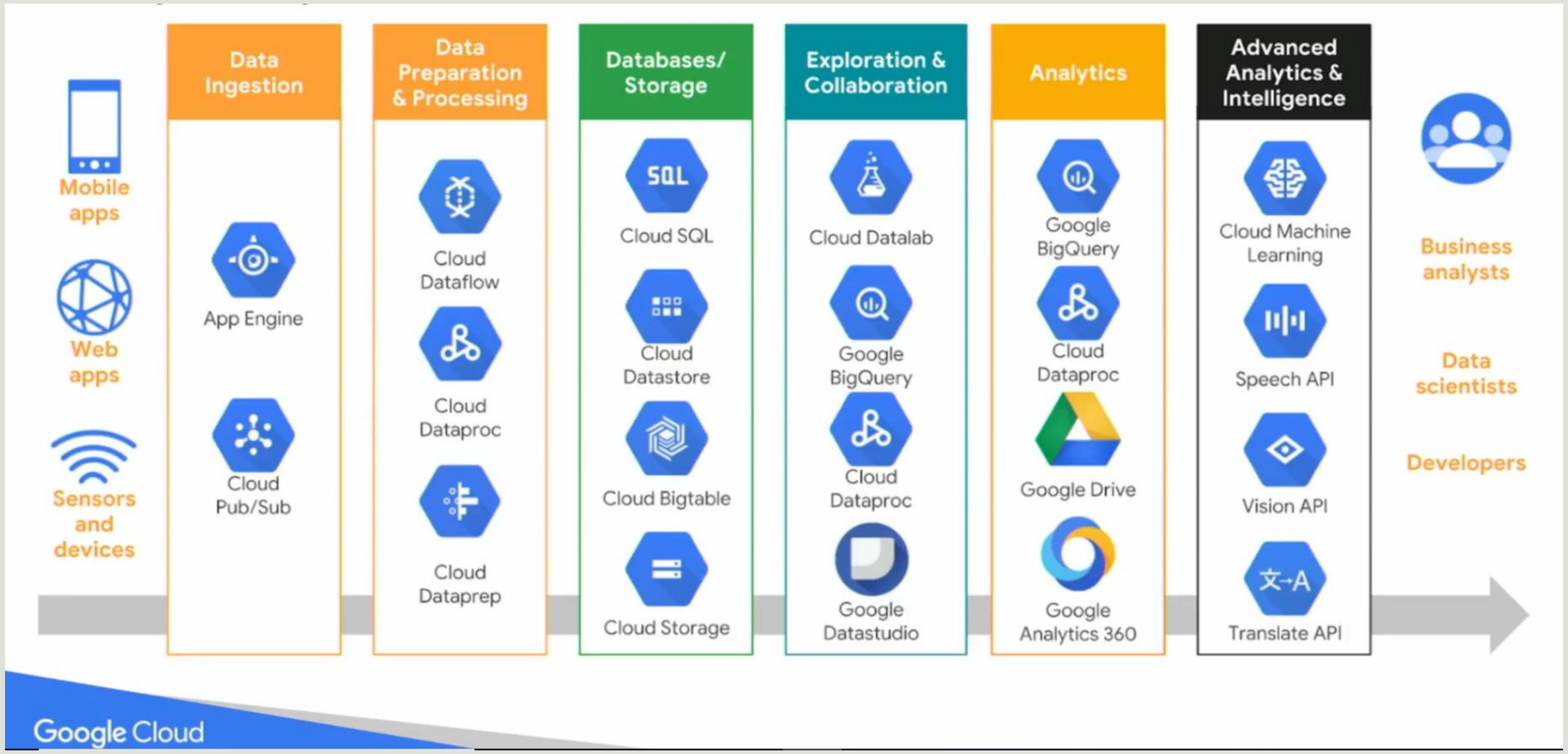
Beyond Data Lakes: Data Platforms

- Data lake has initially conceived as data repositories but it was soon realized that without processing and management tools the usefulness of the data is limited and there is a risk of transforming the data lake into a data swamp

- Due to the progressive broadening of its definition and tot he blurring of the architectural borderlines, the term DL is often replaced by the more general term **data platform** or even **data ecosystem**.

- Today cloud data platforms, such as Google and Amazon, provide several tools for data ingestion, transformation, and analysis. These tools relieve users from the technological complexity of administration by providing additional services that enable companies to focus on functional aspects.

Beyond Data Lakes: Data Platforms



On Premise vs Cloud

- Cloud computing is a **Delivery Model** for on-demand access to a shared pool of configurable computing resources (networks, servers, storage, database, and software) that can be rapidly provisioned and deployed with minimal effort.

- Cloud computing is a computing model that lets you access of software and hardware resources over the Internet on demand and pay-per-use. It enables companies to consume a computer resource as a utility just like electricity rather than having to build and maintain computing infrastructures **in house**.

Cloud Offering Services

Software as a Service (**SaaS**): Consumer uses provider’s applications running on provider's cloud infrastructure.

- Salesforce, Gmail, Facebook

Platform as a Service (**PaaS)** :Consumer can create custom applications using programming tools supported by the provider and deploy them onto the provider's cloud infrastructure.

- Google App Engine, Microsoft Azure CosmosDB, Amazon Dynamo DB

Infrastructure as a Service(**IaaS**): Consumer can use computing resources within provider's infrastructure upon which they can deploy and run arbitrary software, including OS and applications.

- Amazon EC2

Cloud Strengths

Scalability: resource is available as and when the client needs it and, therefore, there are no delays in expanding capacity or the wastage of unused capacity.

No investment in hardware: everything is setup and maintained by the cloud provider, saving the time and cost of doing soon the client side.

Pay for what you use: if the service is only needed for a limited period then it is only paid for over that period and subscriptions can usually be halted at any time.

Updates and Disaster Recovery are automated: Updates will usually be free of charge and deployed automatically by the cloud provider.

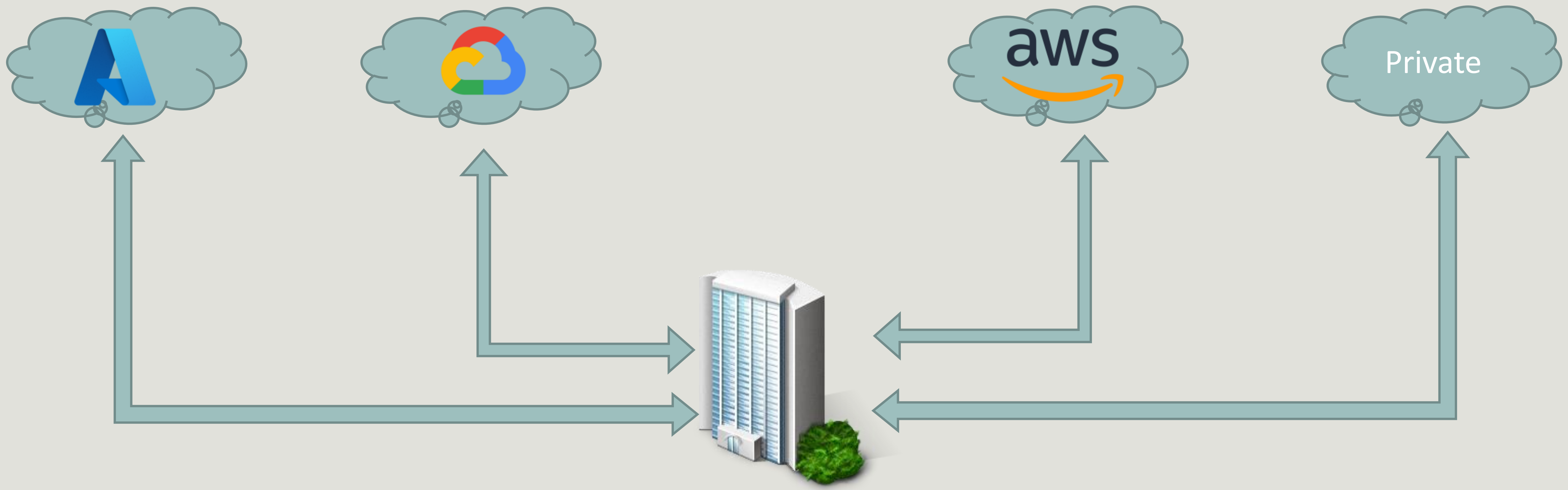
If you budget on the bare resource costs cloud computing is by far more expansive than in house computing. You must budget on the **Total Cost Ownership** (TCO)

More Cloud Strengths

- On demand and pay per use
- Flexibility and Scalability(very quickly)
- Accessibility (any time and anywhere on any web browser or on any pc/mobile device)
- Capital-expenditure Free (without hardware and locally installed software)
- Automatic software updates
- Increased collaboration (edit and share documents anytime, from anywhere)
- Work from anywhere
- Security

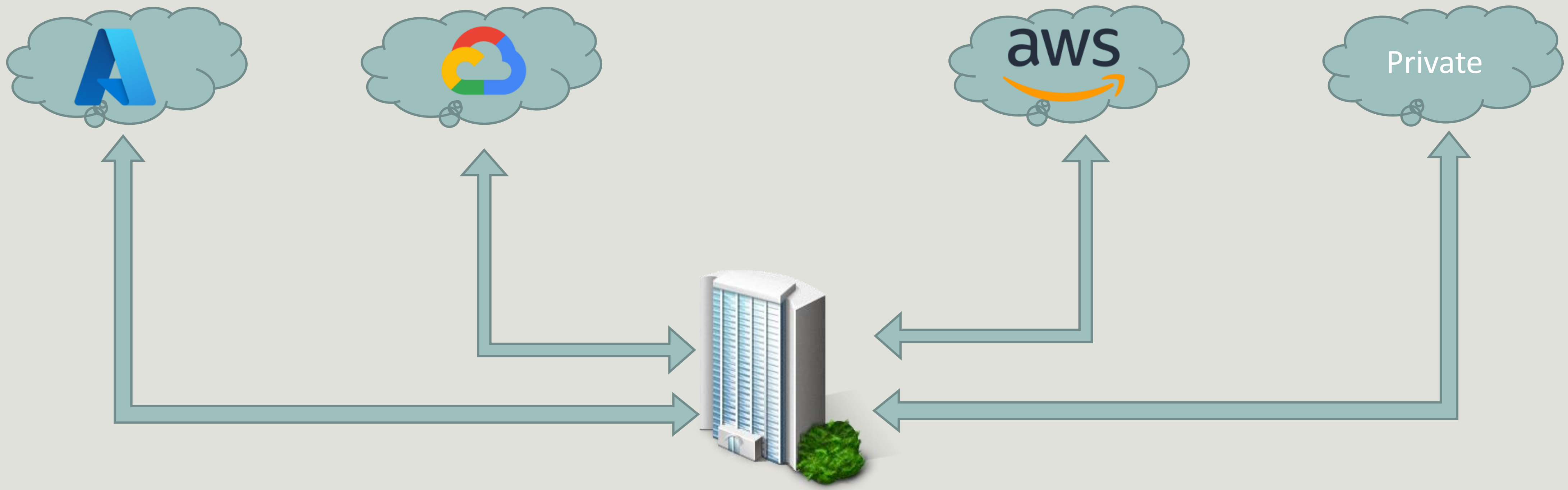
Multi-Cloud Architectures

- In a multi-cloud architecture applications and processes span on different cloud platforms



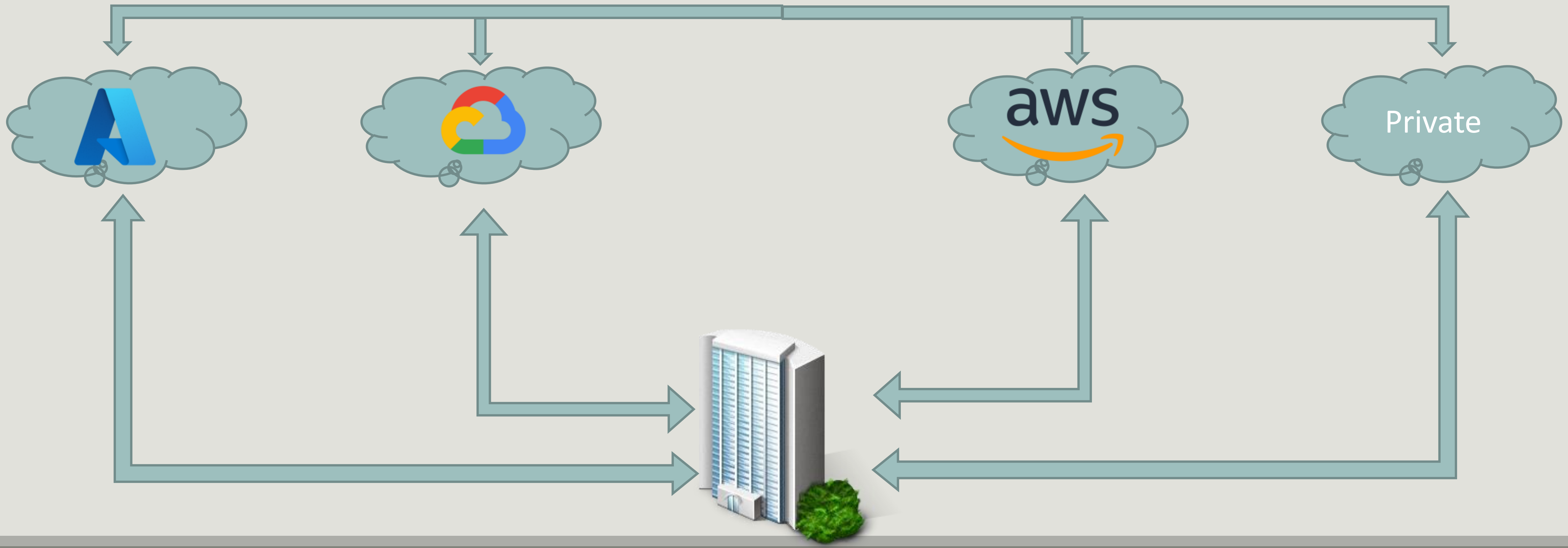
Why a Multi-Cloud Architecture

- Opportunity: cost, service level, by project
- Avoid vendor lock-in and take advantage of best-of-breed solultions (Gartner)
- Risk mitigation: deploying critical systems across multiple cloud services provides additional fault tollerance



Multi-Cloud Architectures Challenges

- Security and Trust
- Deploying, Balancing & Provisioning



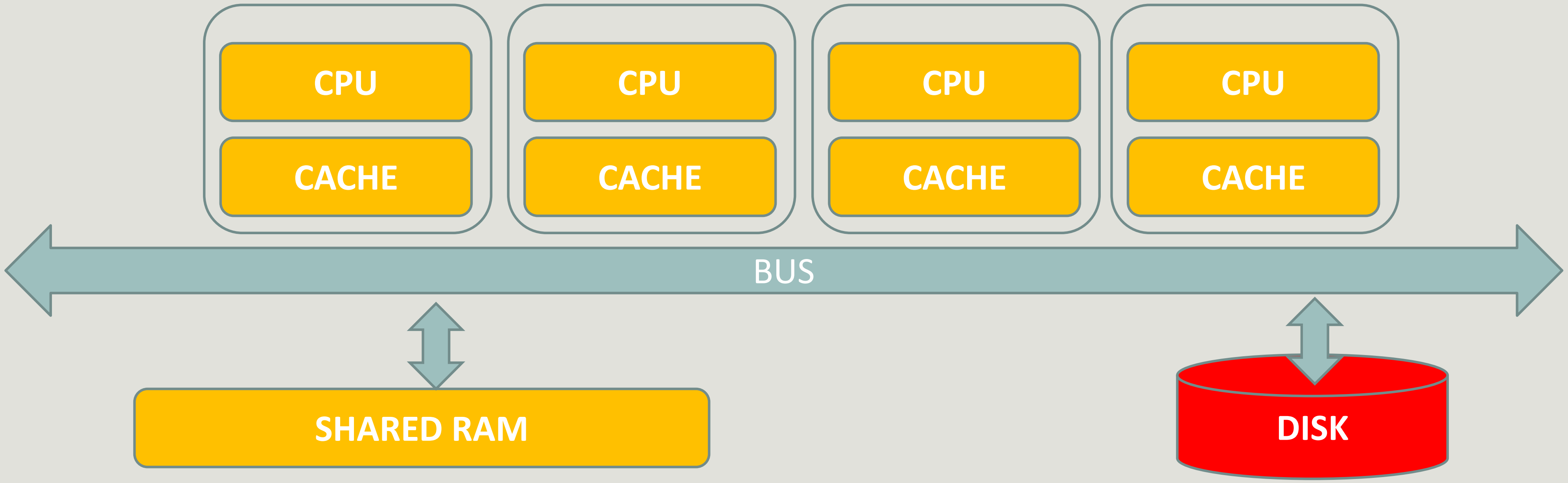
Hadoop

The Apache Hadoop software library is a framework that allows for the **distributed processing** of **large data sets** across **clusters of computers** using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver **high-availability** and **reliability**, the library itself is designed to **detect and handle failures at the application layer**, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

SMP Architecture

Traditional RDBMS are based on *Symmetric Multi Processing* (SMP) architecture: several processors share the same RAM, the same I/O bus and the same disk/disk array

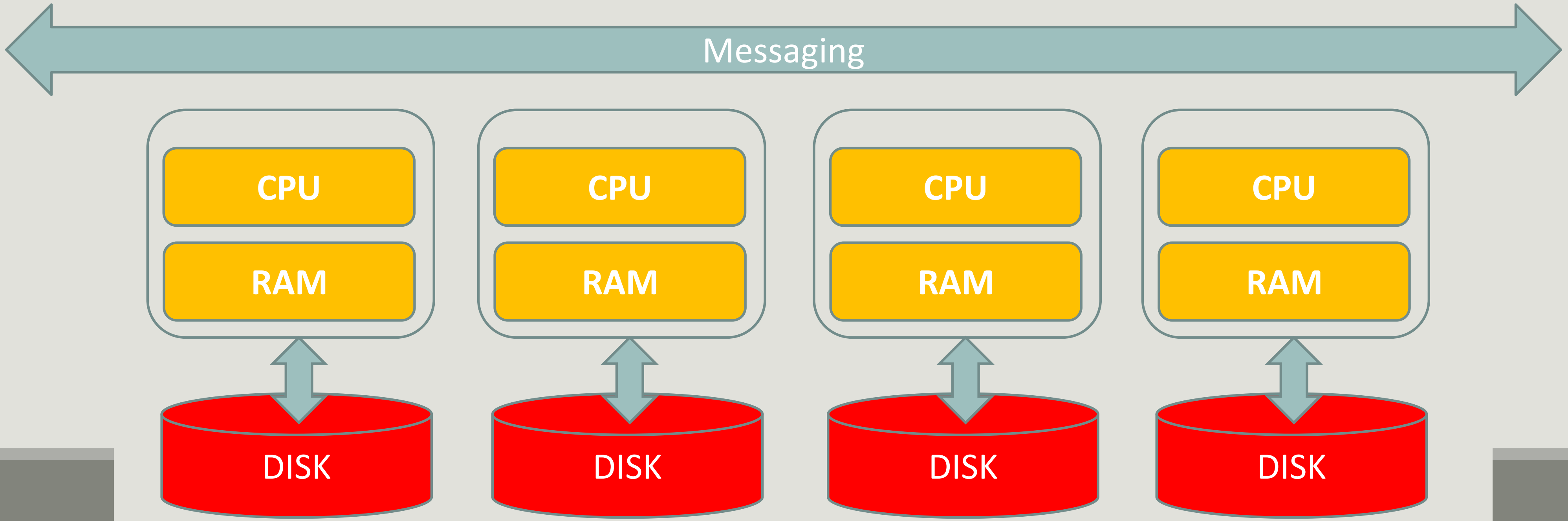
- The limit of such architecture depends on the physical number of devices that can be mounted and by the BUS bottleneck



MPP Architecture

In a *Massively Parallel Processing* (MPP) Architecture several processors, each equipped with its own RAM and disks, collaborate to solve a single problem by splitting it in several independent tasks. It is also called shared nothing architecture

- Examples of MPP appliances are Teradata, Nettezza that are mainly used for Data Warehouse applications



Hadoop vs MPP

In MPP, as in MapReduce, processing of data is distributed across a bunch of compute nodes, these separate nodes process their data in parallel and the node-level output sets are assembled together to produce a final result set, but:

- MPP systems are based on **high-level HW and proprietary SW** while Hadoop is based on **commodity HW and open source SW**
- Hadoop is natively controlled through imperative code while MPP appliances are queried though declarative query.
- In a great many cases, SQL is easier and more productive than is writing MapReduce jobs, and database professionals with the SQL skill set are more plentiful and less costly than Hadoop specialists.

Since they share the same computation mechanism Hadoop and MPP systems are relatives and there is no reasons for the two approaches to converge. Some specific solutions, such as Cloudera Impala, are examples of such process.

Grid Computing Architecture

Grid computing is the collection of computer resources from multiple locations to reach a common goal. Grid computing is distinguished from conventional high performance computing systems such as cluster computing in that:

- Grid computers have each node set to perform a different task/application.
- Grid computers tend to be more heterogeneous and geographically dispersed (thus not physically coupled) than cluster computers.
- Although a single grid can be dedicated to a particular application, commonly a grid is used for a variety of purposes.
- Grids are often constructed with general-purpose grid middleware software libraries.

HPC Architecture

The term High-Performance Computing refers to massively parallel system specifically devoted to solve cpu-intensive tasks:

- HPC is often used as a synonym of Super Computer even if supercomputer is a more general term referring to a system that is at the frontline of contemporary processing capacity. Its architecture and features can change along time
- HPC main applications are scientific simulations, weather forecasts, 3d modeling, etc.
- In many cases the processors share the same set of disk drive

Scalability

Scalability is ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.

Methods of adding more resources for a particular application fall into two broad categories: horizontal and vertical scaling.

- To **scale horizontally** (or **scale out**) means to add more nodes to a system, such as adding a new computer to a distributed software application.
- To **scale vertically** (or **scale up**) means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer.

Vertical scalability is typically limited by the actual capability of the architecture to mount more resources and by the physical limitation of such resources (e.g. processor speeds). Nowadays, the availability of low cost commodity hardware and network connections make scaling out often more convenient than scale up.

NoHadoop: Not only Hadoop

The data science toolkit is composed by several other technologies for data storing and handling that are born around the concept *One size does not fit all*

NoSQL – Not Only SQL

- Mongo DB – Document Oriented
- Neo4J – Graph-Based
- Hbase, BigTable – tabular
- Redis – Key-Value

NewSQL (support ACID properties)

- VoltDB

Text analytics

- Elastic Search
- Oracle Endeca (in memory)

Appliance

- Oracle Exadata & Exalytics (in memory)
- SAP HANA (in memory)
- Teradata Aster
- SAS LASR Analytic Server (in memory)

Data Analysis

- Alteryx (Data blending)
- R (statistical analysis and data mining)
- Tableau (Extemporary and Exploratory analysis)
- Brandwatch, Tracx (Social Media Monitoring)

Hadoop1 vs Hadoop2

- **Hadoop1** (i.e. map-reduce) was born to cope with very specific tasks
  - Difficult to program
  - Robust and efficient on huge batch jobs but not suited for on-line processing

- **Hadoop2** (i.e. Yarn) is born to makes Hadoop a *more* general purpose programming framework
  - Map-reduce is just one of the possible programming paradigms
  - Support for SQL
  - Suited for on-line processing
- The community itself is trying to make it easier to non-Hadoop experts to use it
  - Graphical control consoles
  - Connectors to many external software (e.g. OLAP Microstrategy)

Native Big Data Application

Traditional Database / Data Analysis Application

Distributions and the cloud

In many cases Hadoop is not exploited from an owned cluster, but rather from a cloud provider. Several factors impacts on the choice of the provider :

- Costs
- Completeness of the cloud Market place
- Other projects hosted on the cloud and producing the source data (moving big data may be expansive!)
- ….

Thus the distribution choice can be either a driver or a consequence

MYTHBUSTERS

A disenchanted analysis of Hadoop

A disenchanted analysis of Hadoop

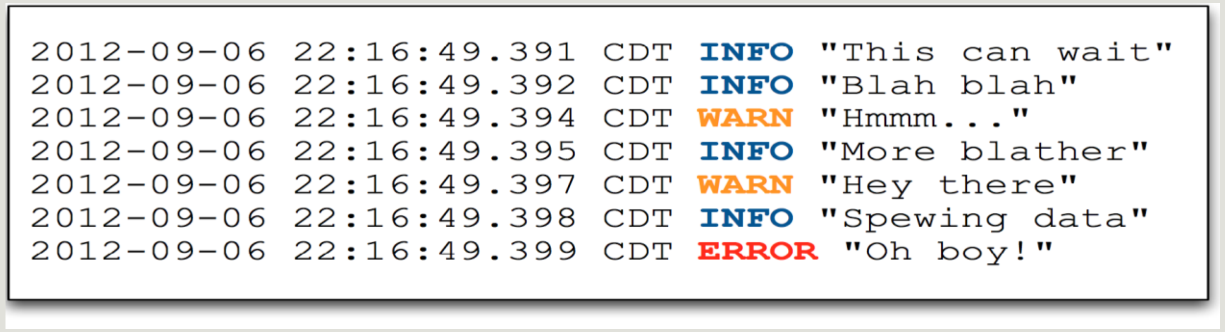
- Hadoop is a distributed system but with respect to previous distributed architectures it makes cluster computing easier to administer, and easier to program. It is based on commodity HW and open source SW
- Map-reduce is a programming model that makes it simple carrying out parallel computations but, NOT all the problems/algorithms can be efficiently parallelized on such architecture. Others are not parallelizable at all.
    - A lot of research/development is running on porting algorithms to Hadoop
- Hadoop is born in the Era of cloud computing: ***cloud service providers are pushing Hadoop*** and Big Data since they require a lot of computational power, a lot of disk storage that can be bought in the cloud
- Hadoop is not more efficient than centralized architectures (e.g. ORACLE DBMS) or Appliances (e.g. Teradata) but it can ***scale out*** at a lower price

A disenchanted analysis of Hadoop

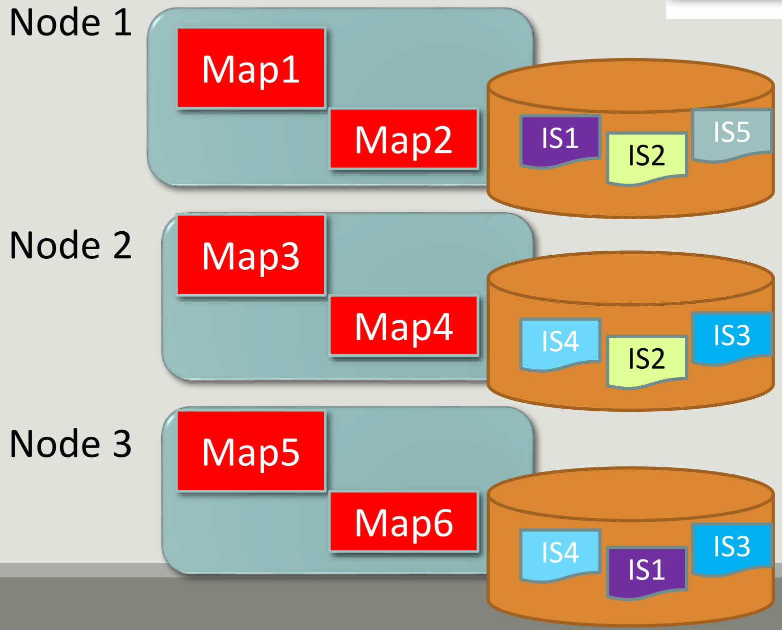
- Hadoop is the answer of programmers companies (i.e. Google) to data modeling, storing and handling
    - Favors programming to modeling if compared to traditional data base systems
    - Requires a radical shift in the way data are modeled and transformed
    - It imposes a programmer paradigm with a limited set of functionalities

The Map-Reduce paradigm: an example

Counting the number of messages of different types

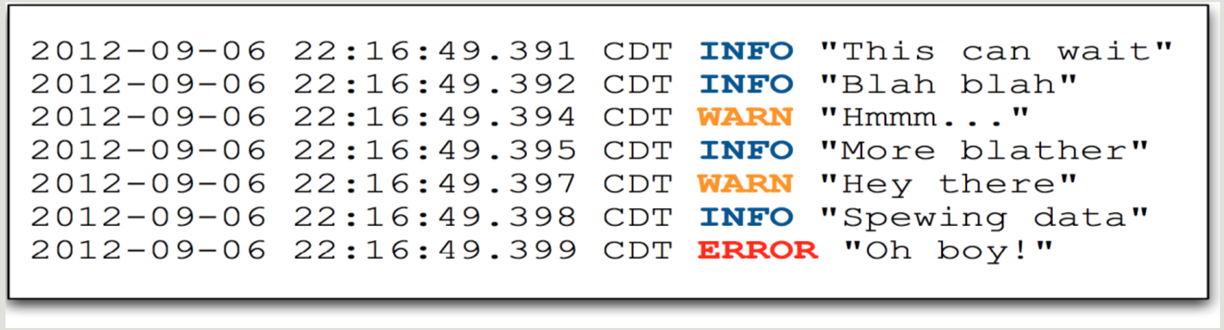


```
2012-09-06 22:16:49.391 CDT INFO "This can wait"
2012-09-06 22:16:49.392 CDT INFO "Blah blah"
2012-09-06 22:16:49.394 CDT WARN "Hmmm..."
2012-09-06 22:16:49.395 CDT INFO "More blather"
2012-09-06 22:16:49.397 CDT WARN "Hey there"
2012-09-06 22:16:49.398 CDT INFO "Spewing data"
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```

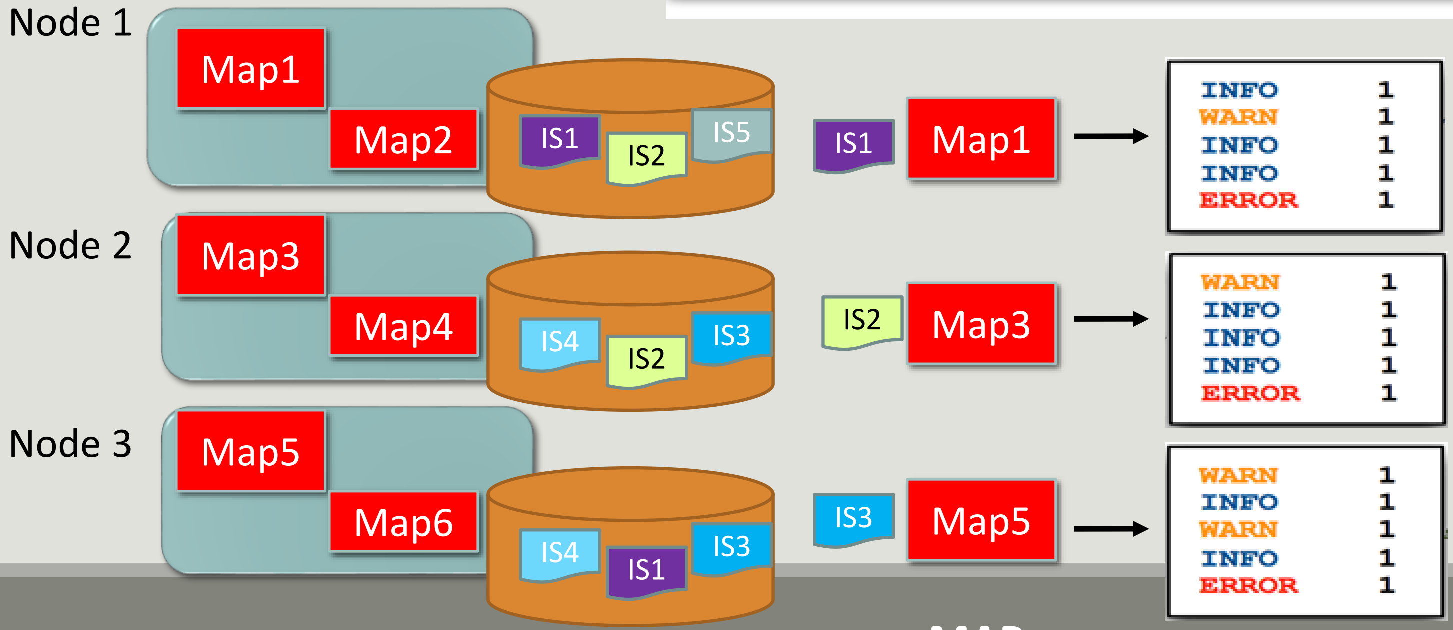


The Map-Reduce paradigm: an example

Counting the number of messages of different types



```
2012-09-06 22:16:49.391 CDT INFO "This can wait"
2012-09-06 22:16:49.392 CDT INFO "Blah blah"
2012-09-06 22:16:49.394 CDT WARN "Hmmm..."
2012-09-06 22:16:49.395 CDT INFO "More blather"
2012-09-06 22:16:49.397 CDT WARN "Hey there"
2012-09-06 22:16:49.398 CDT INFO "Spewing data"
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```



```
INFO   1
WARN   1
INFO   1
INFO   1
ERROR  1
```

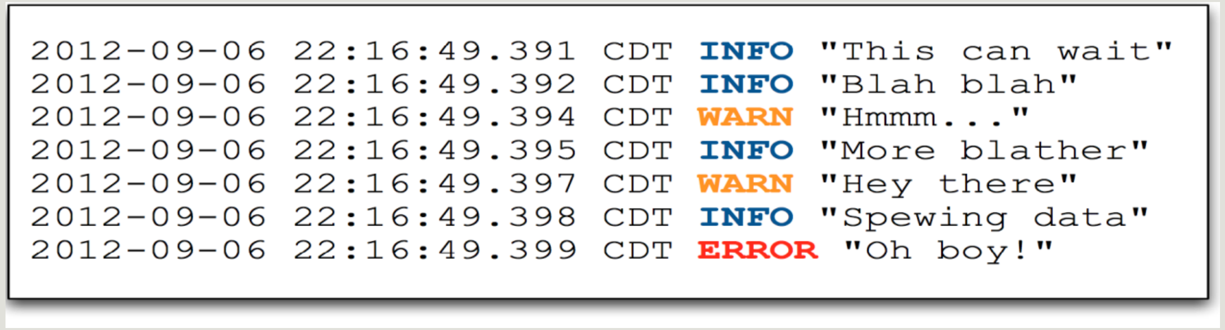
```
WARN   1
INFO   1
INFO   1
INFO   1
ERROR  1
```

```
WARN   1
INFO   1
WARN   1
INFO   1
ERROR  1
```

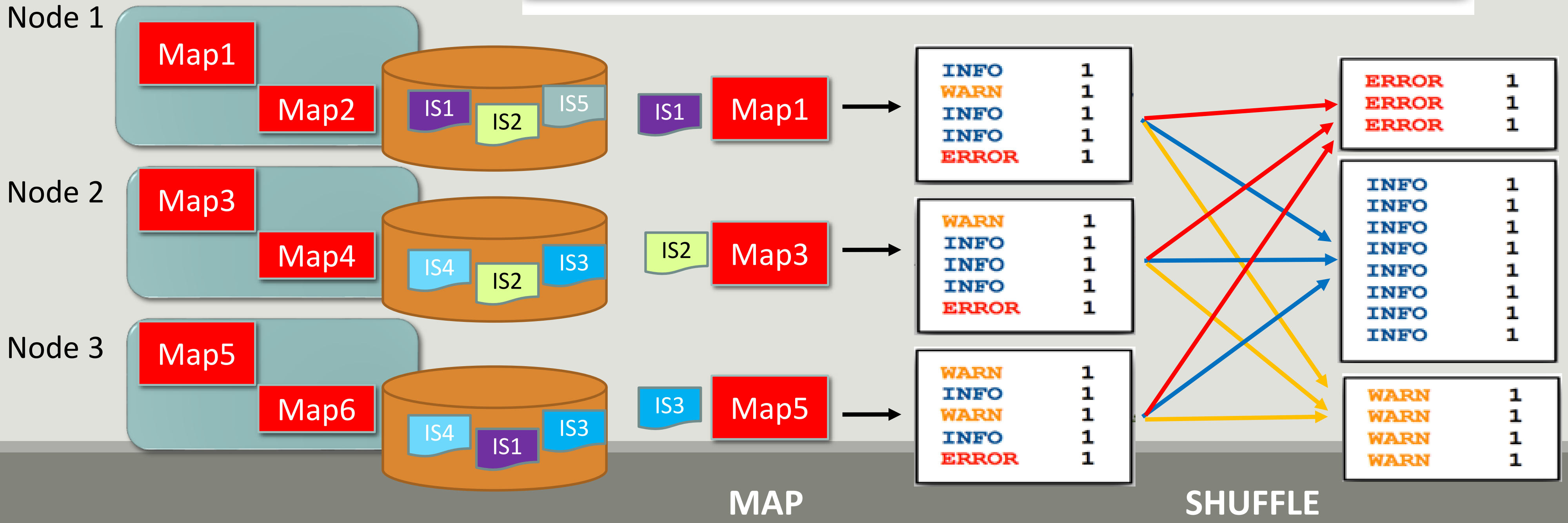
MAP

The Map-Reduce paradigm: an example

Counting the number of messages of different types



```
2012-09-06 22:16:49.391 CDT INFO "This can wait"
2012-09-06 22:16:49.392 CDT INFO "Blah blah"
2012-09-06 22:16:49.394 CDT WARN "Hmmm..."
2012-09-06 22:16:49.395 CDT INFO "More blather"
2012-09-06 22:16:49.397 CDT WARN "Hey there"
2012-09-06 22:16:49.398 CDT INFO "Spewing data"
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```



```
INFO   1
WARN   1
INFO   1
INFO   1
ERROR  1
```

```
WARN   1
INFO   1
INFO   1
INFO   1
ERROR  1
```

```
WARN   1
INFO   1
WARN   1
INFO   1
ERROR  1
```

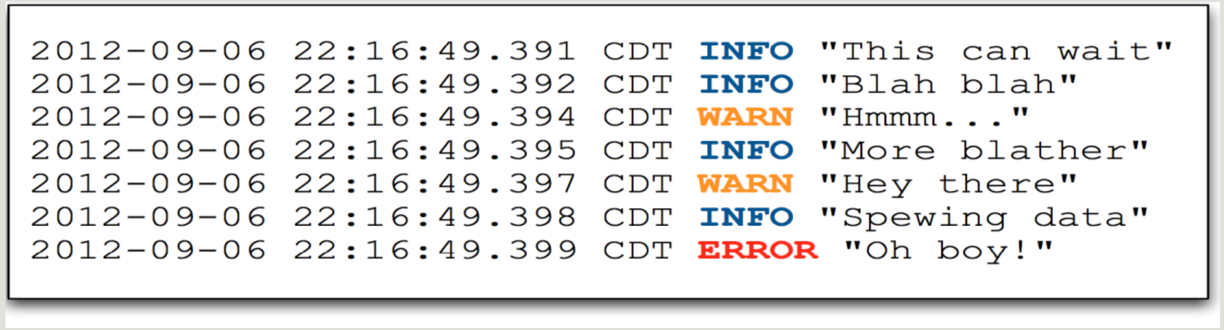
```
ERROR  1
ERROR  1
ERROR  1
```

```
INFO   1
INFO   1
INFO   1
INFO   1
INFO   1
INFO   1
INFO   1
INFO   1
```

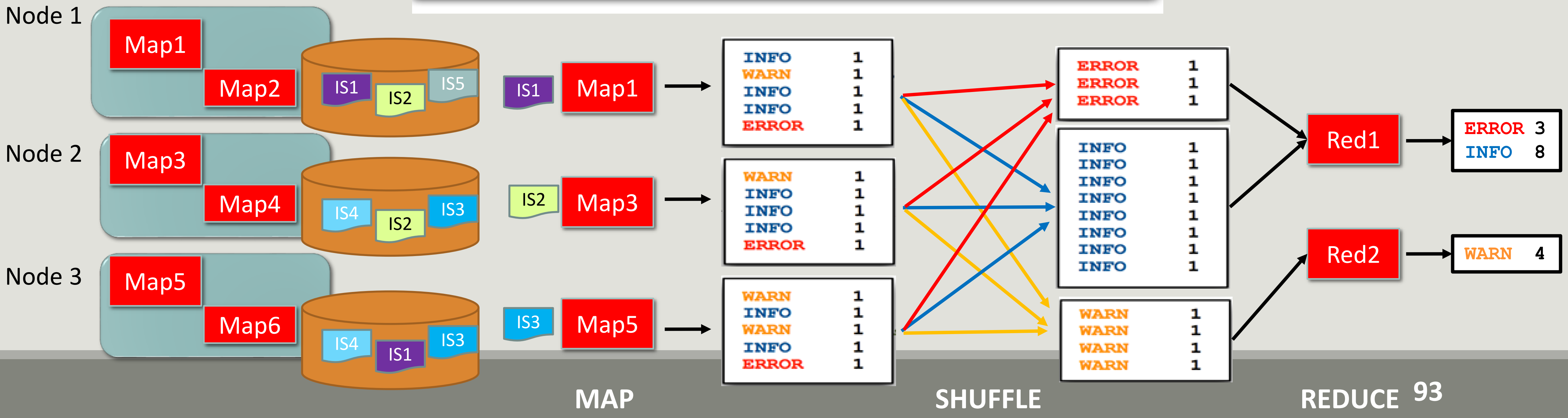
```
WARN   1
WARN   1
WARN   1
WARN   1
```

The Map-Reduce paradigm: an example

Counting the number of messages of different types



```
2012-09-06 22:16:49.391 CDT INFO "This can wait"
2012-09-06 22:16:49.392 CDT INFO "Blah blah"
2012-09-06 22:16:49.394 CDT WARN "Hmmm..."
2012-09-06 22:16:49.395 CDT INFO "More blather"
2012-09-06 22:16:49.397 CDT WARN "Hey there"
2012-09-06 22:16:49.398 CDT INFO "Spewing data"
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```



Hadoop 1 vs Hadoop 2

The Hadoop 1.0 project is around since ten years. Its main features are:

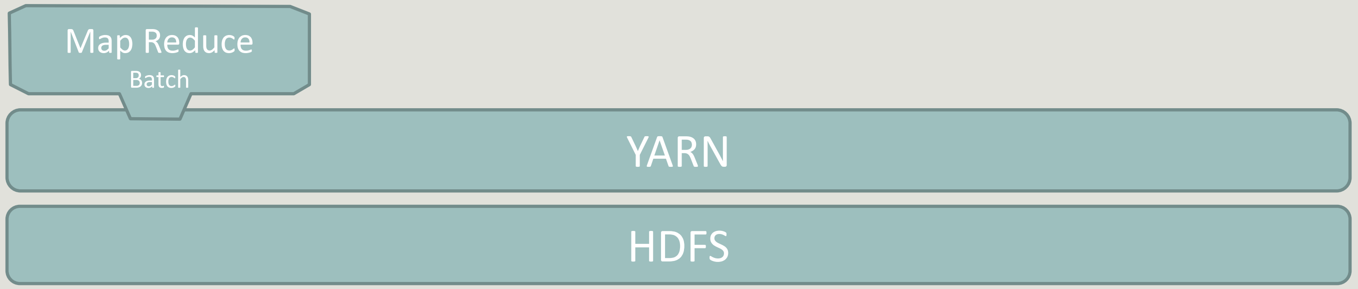
- It was inspired by a similar Google project proposing Map-Reduce computation framework and the proprietary Google File System
- It is mainly oriented to execute batch Map-Reduce

The first stable Hadoop 2.0 release is dated 2013. Its main features are:

- It solves some bottlenecks and single-point of failure of Hadoop 1
- it turns Hadoop to be a **Data Operating system** by adding **YARN** (*Yet Another Resource negotiator*) a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications
- Its opens to computation mechanisms different from Map-Reduce

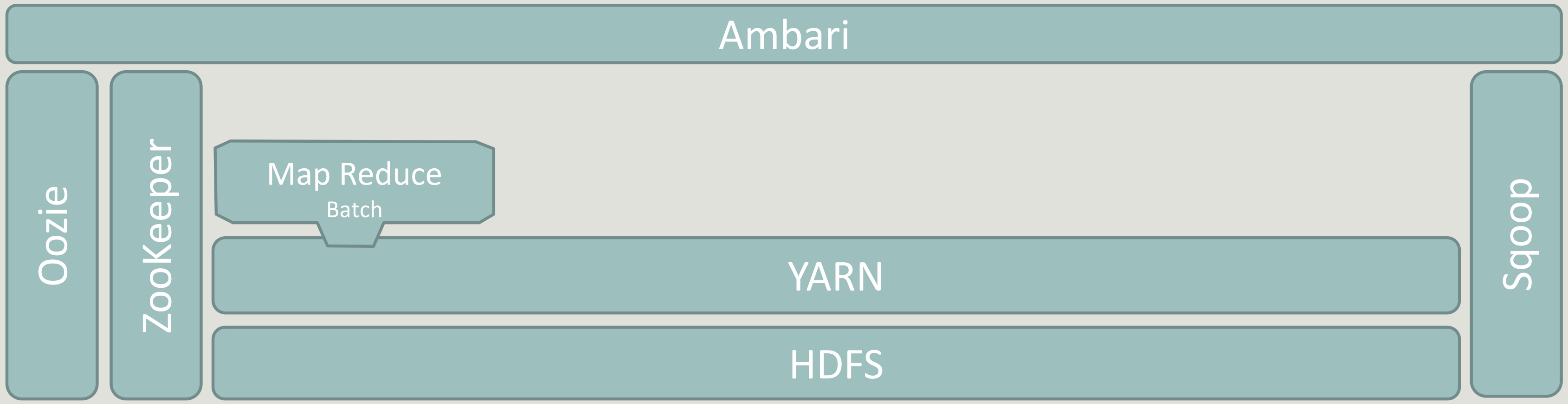
Main modules

Common: the common utilities that support the other Hadoop modules
Distributed File System: a distributed file system that provides high-throughput access to application data
YARN: A framework for job scheduling and cluster resource management
Map Reduce: A YARN-based system for parallel processing of large data sets



More modules

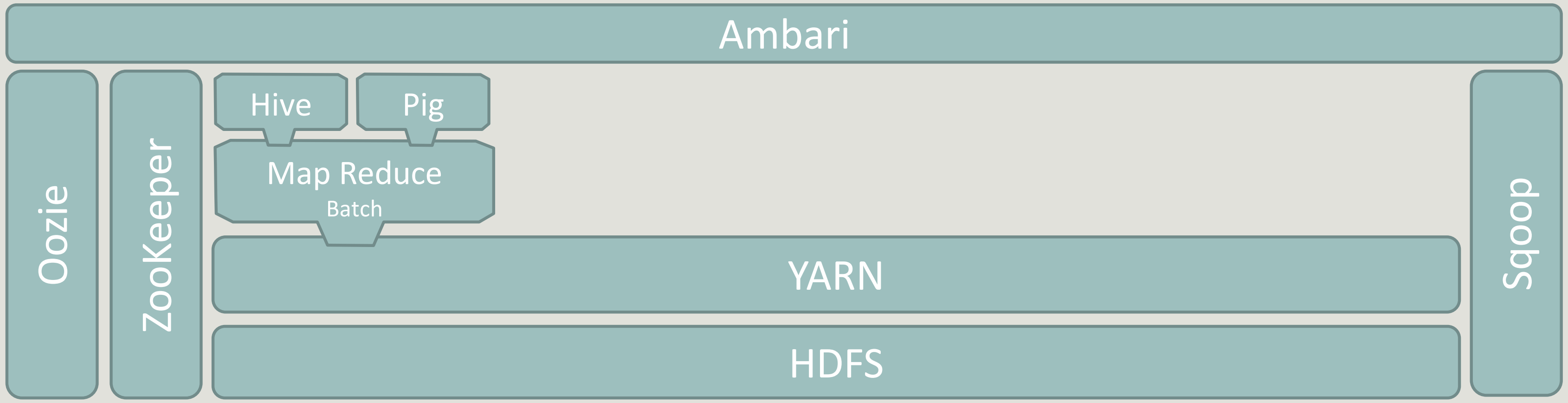
Ambari: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop.
ZooKeeper: A high-performance coordination service for distributed applications.
Oozie is a workflow scheduler system to manage Apache Hadoop jobs. Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions.
Sqoop is a tool designed for efficiently transferring bulk data between Hadoop and structured datastores such as relational databases.



and more modules…

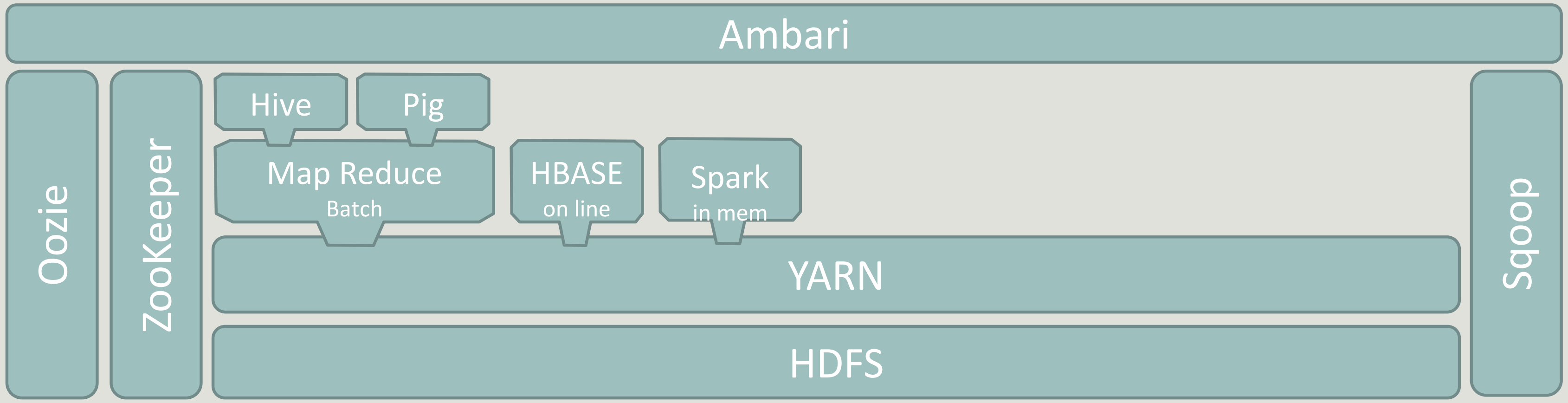
Pig: a platform for analyzing large data sets that consists of a high-level language (Pig-Latin) for expressing data analysis programs. Pig-Lating programs look like ETL flows that transform data. Pig-Latin programs are compiled to Map-Reduce jobs.
Hive: provides a mechanism to project structure onto HDFS data and query the data using a SQL-like language called HiveQL. HiveQL also allows traditional map-reduce programmers to plug in their custom mappers and reducers to improve preformances.

Since Pig code and hive queries are implemented as map-reduce jobs, they determine a batch execution.



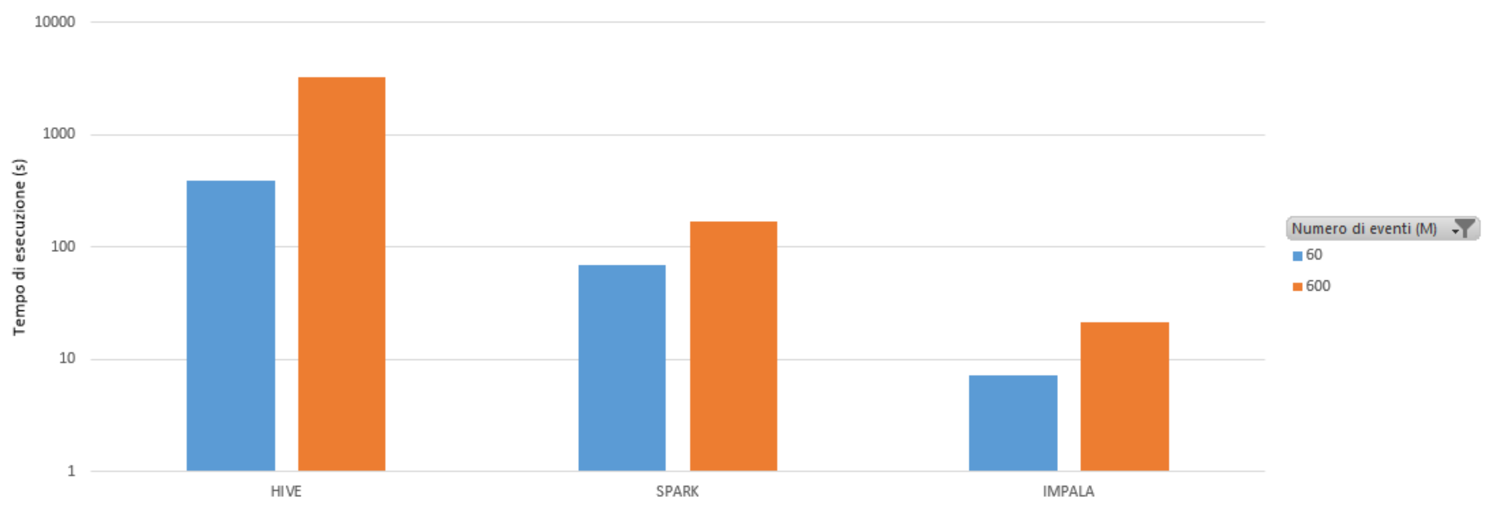
and more modules…

HBase: is a non-relational, distributed DBMS. It runs on top of HDFS providing BigTable-like capabilities for Hadoop. HBase features compression, in-memory operation, and Bloom filters on a per-column basis, it allows flexible schemata. Basically HBASE is a key/value data store.
Spark: In contrast to two-stage disk-based Map-Reduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well suited to machine learning algorithms.

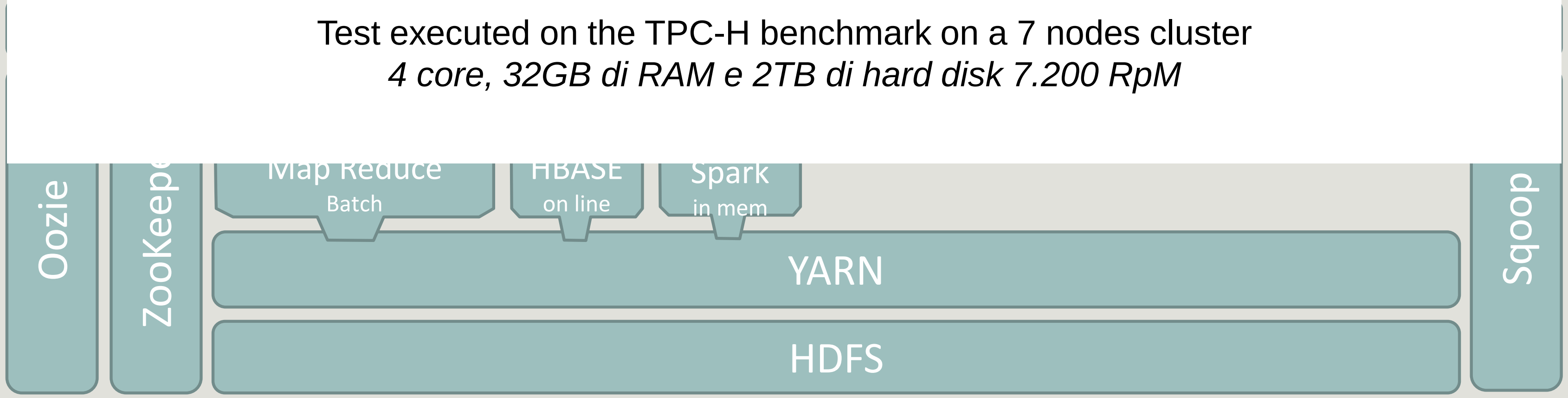


and more modules...

HBase: is … -like
capabilit … n filters
on a per- … store.
Spark: In
primitive … wing user
program … ited to
machine



Test executed on the TPC-H benchmark on a 7 nodes cluster
4 core, 32GB di RAM e 2TB di hard disk 7.200 RpM



Spark

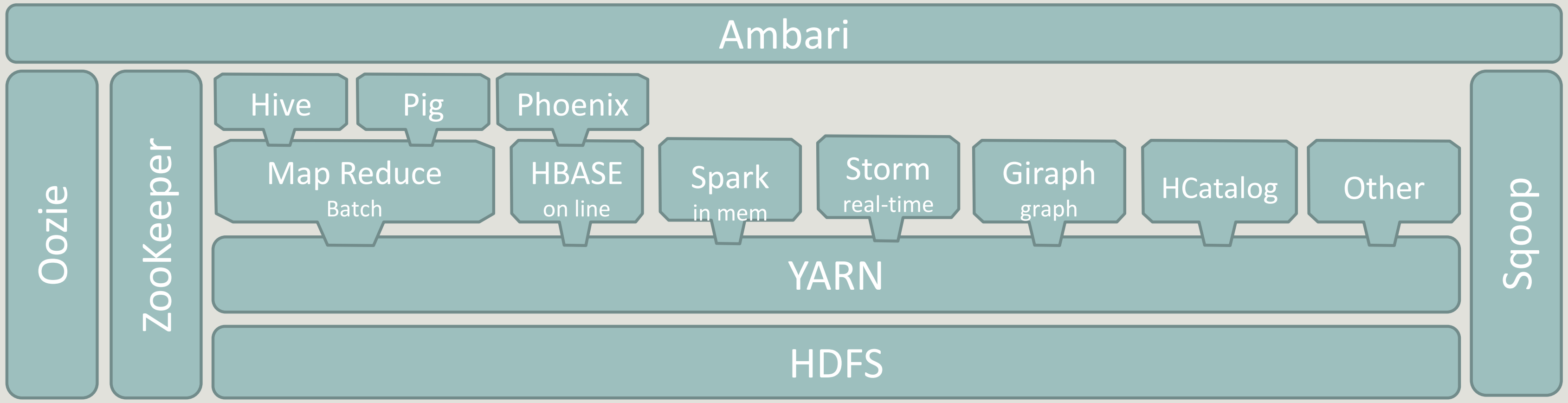
Spark was initially started at UC Berkeley AMPLab in 2009. It requires a cluster manager and a distributed storage system. For **cluster manager**, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos. For **distributed storage**, Spark can interface with a wide variety, including HDFS, Cassandra, and Amazon S3.

The Spark project consists of multiple components

- **Spark SQL** introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data. Spark SQL provides a domain-specific language to manipulate SchemaRDDs in Scala, Java, or Python. It also provides SQL language support, with command-line interfaces and ODBC/JDBC server
- **Spark Streaming** leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD transformations on those mini-batches of data.
- **MLlib** is a distributed machine learning framework on top of Spark that because of the distributed memory-based Spark architecture is ten times as fast as Hadoop disk-based Apache Mahout.
- **GraphX** is a distributed graph processing framework on top of Spark. It provides an API for expressing graph computation that can model the Pregel abstraction.

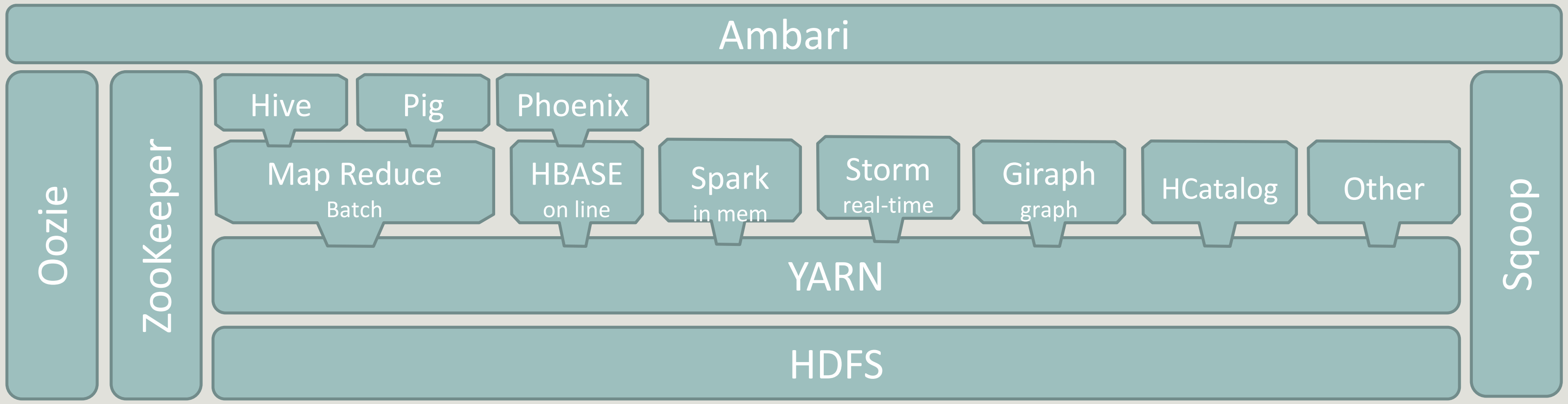
… and more modules

Storm: a distributed real-time computation system for processing large volumes of high-velocity data. Storm on YARN is powerful for scenarios requiring real-time analytics, machine learning and continuous monitoring of operations.
Giraph: an iterative graph processing system built for high scalability. For example, it is currently used at Facebook to analyze the social graph formed by users and their connections.
HCatalog create a relational abstraction of data in HDFS and ensures that users need not worry about where or in what format their data is stored. HCatalog displays data from RCFile format, text files, or sequence files in a tabular view.



… and more modules

Phoenix: is a relational database layer over HBase delivered as a client-embedded JDBC driver targeting low latency queries over HBase data. Apache Phoenix takes your SQL query, compiles it into a series of HBase scans, and orchestrates the running of those scans to produce regular JDBC result sets.



Research Thesis
Research Fellowships
Both on Research and Professional projects
PhD Positions
Related to BI and Data mining issues on Big Data

Business Intelligence Group

Hadoop Distributed File System– HDFS1

The HDFS is a distributed file system designed to run on commodity hardware.

- Hardware failure is the norm rather than the exception. Therefore, **detection of faults and quick, automatic recovery** from them is a core architectural goal of HDFS.
- Applications that run on HDFS need streaming access to their data sets. HDFS is designed more for **batch processing rather than interactive use by users**. The emphasis is on high throughput of data access rather than low latency of data access.
- Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to **support large files**.
- HDFS applications need a **write-once-read-many** access model for files. A file once created, written, and closed need not to be changed. This assumption simplifies data coherency issues and enables high throughput data access.
- A computation requested by an application is much more efficient if it is executed near the data it operates on. HDFS provides interfaces for applications to move themselves closer to where the data is located (**data locality**).

Hadoop Distributed File System– HDFS 2

With YARN in HDP 2.0, new applications are emerging that will execute on the same Hadoop cluster against data in HDFS. This range of applications have different data access patterns and requirements, going beyond just batch

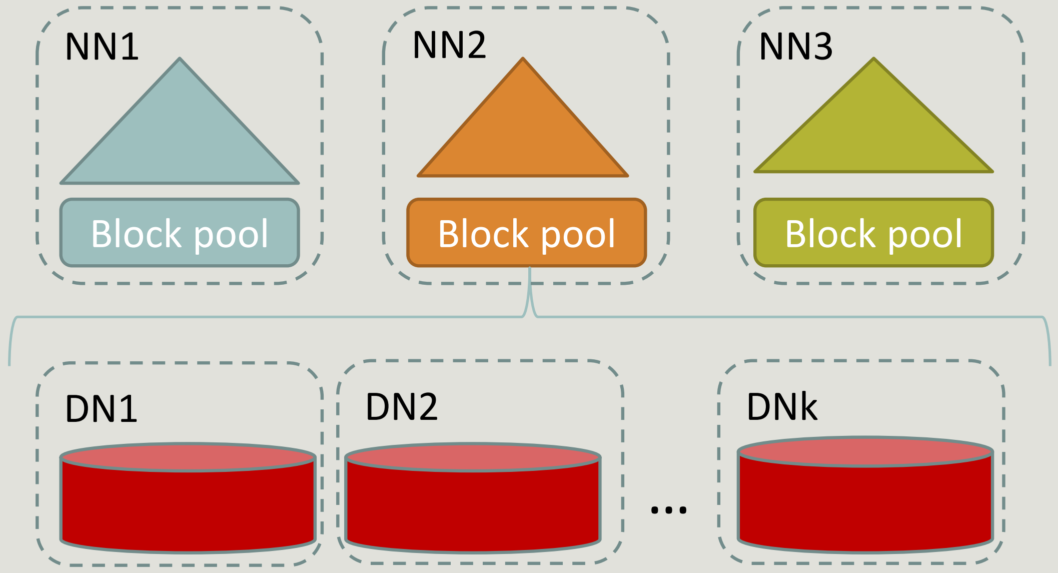
While in HDFS 1 all disks were treated equally in each node **HDFS 2 is technology aware** to take advantage of all storage and memory hardware – spinning disks, solid state drives, RAM memory and external storage

The cluster system administrator will be able to **configure the storage media available** on each node. HDFS will then allow datasets to be given a storage tier preference. Applications will be able to specify a **Storage Medium preference** when creating files that supports the applications' read work loads.

HDFS Federation

Differently from HDFS 1, HDFS 2 implements a federation of name spaces.

- A **name space** (NS) is a hierarchy of files and directories (i.e. two name spaces can have the same file name in the same directory). Metadata for each name space are stored on a **Name Node** (NN).
- The NN maintains the name space tree and the mapping of **data blocks** to **Data Nodes** (DN).
- Name spaces use blocks grouped under a **Block Pool**. A DN can provide blocks to more than one name space.



HDFS Federation

Scalability: because the NN keeps all the name space and block locations in memory, the size of the NN heap limits the number of files and also the number of blocks addressable. This also limits the total cluster storage that can be supported by the NN.
Performance: NN is the single point for storage and management of meta-data, it can become a bottleneck for supporting a huge number of files, especially a large number of small files.
Availability: you can separate name spaces of different applications improving the overall availability of the cluster.
Maintainability, Security & Flexibility: block pool abstraction allows other services to use the block storage with perhaps a different name space structure. Each name space is isolated and not aware of the others.

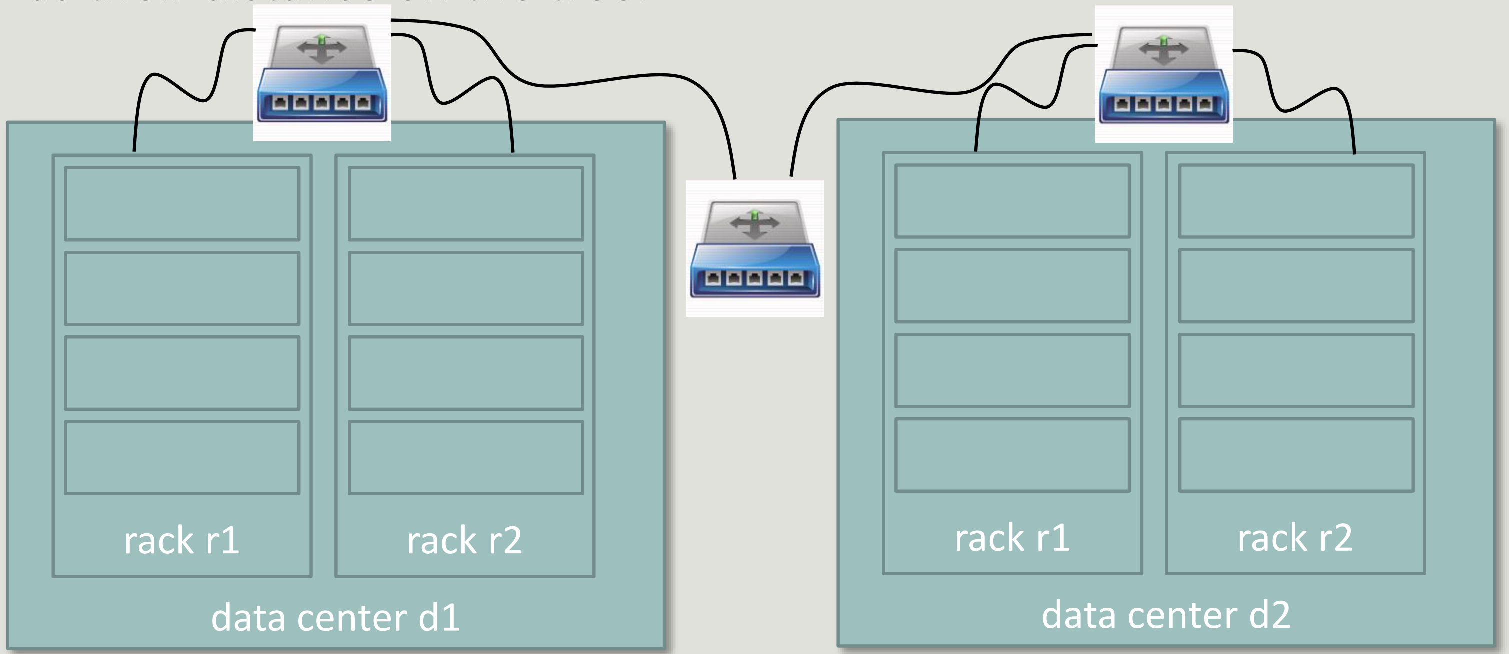
Applications can read/write on more than one name space.

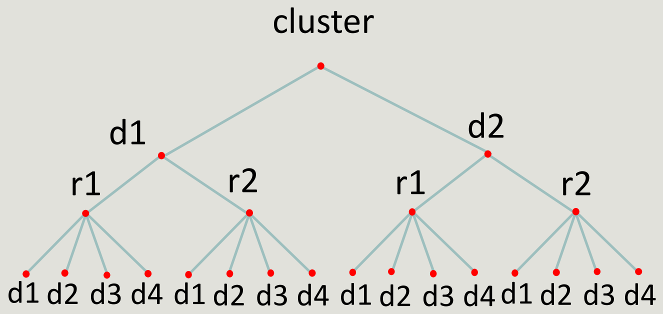
Cluster Topology

In order to carry out proper choices the cluster must be aware of the Cluster Topology that is defined during the cluster setting phase. Block storage and process allocation (data locality) are tasks that need such information

Nodes are organized in racks and racks are organized in data center

Hadoop models such concepts in a tree-like fashion and computes the distance between nodes as their distance on the tree.



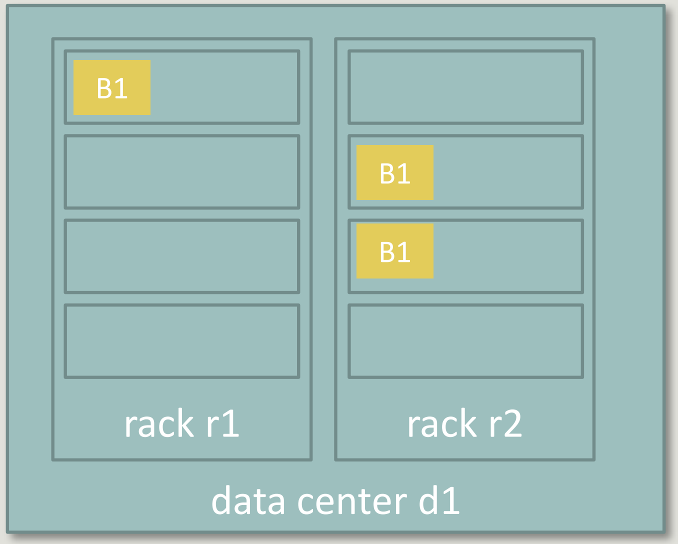


Data Blocks & Data Replica

The file content is split into large blocks (default 128 MB), and each block of the file is independently replicated at multiple Data Nodes in order to improve performance and robustness.
Replication is aware of the cluster topology. For each data block the name node stores the list of data nodes storing it. The default replication factor is 3:

- Copy 1 is stored on the node (n1) where the client issued the write command
- Copy 2 is stored on a node (n2) in a rack (r2) different from the one of n1 (*off-rack*)
- Copy 3 is stored on a node, different from n2, but that belongs to r2

In case of unavailability the system transparently rebalances replicas



Data Blocks VS OS Blocks

The HDFS data block size ranges between 64MB and 1GB

- The data block is the smallest unit of data addressed in the name node
- Large data blocks reduces the cost for handling the data request at the cluster level rather than at the node level. In HDFS, those requests go across a network and come with a lot of overhead: each request has to be processed by the Name Node to figure out where that block can be found.
  - Lets say you have a 1000Mb file. With a 4k data block size, you'd have to make 256,000 requests. If you use 64Mb blocks, the number of requests goes down to 16, greatly reducing the cost of overhead and load on the Name Node.

- Locally on a data node a data block request will be turned in several OS blocks requests
- The OS blocks (i.e. disk pages) composing a data block can be accessed independently
- If a file is smaller than the data block size it will be stored in a smaller file on the disk

Data Locality

Hadoop exploits cluster topology and data block replication to apply the **data locality principle**

When computations involves large set of data its cheaper (i.e. faster) to move code to data rather than data to code

The following cases respect the order the resource manager prefers:

1. Process and data on the same node
2. Process and data on the different node of the same rack
3. Process and data on different racks of the same data center
4. Process and data on different racks of the different data centers

Name Space & Name Node

Name space meta-data, called the **image**, includes:

- The name space tree and the mapping of blocks to **Data Nodes**.
- **Inodes** record attributes like permissions, modification and access times, name space and disk space quotas.

Name Node keeps the entire name space image in RAM. The persistent record of the image stored in the Name Node's local native file system is called a **checkpoint**.

The Name Node records changes to HDFS in a **write-ahead log** called the **journal** in its local native file system

A **secondary name node** regularly connects with the primary one and builds snapshots of the primary name node's directory information, which the system then saves to local or remote directories. These check-pointed images can be used to restart a failed primary name node without having to replay the entire journal of file-system actions.

High-Availability

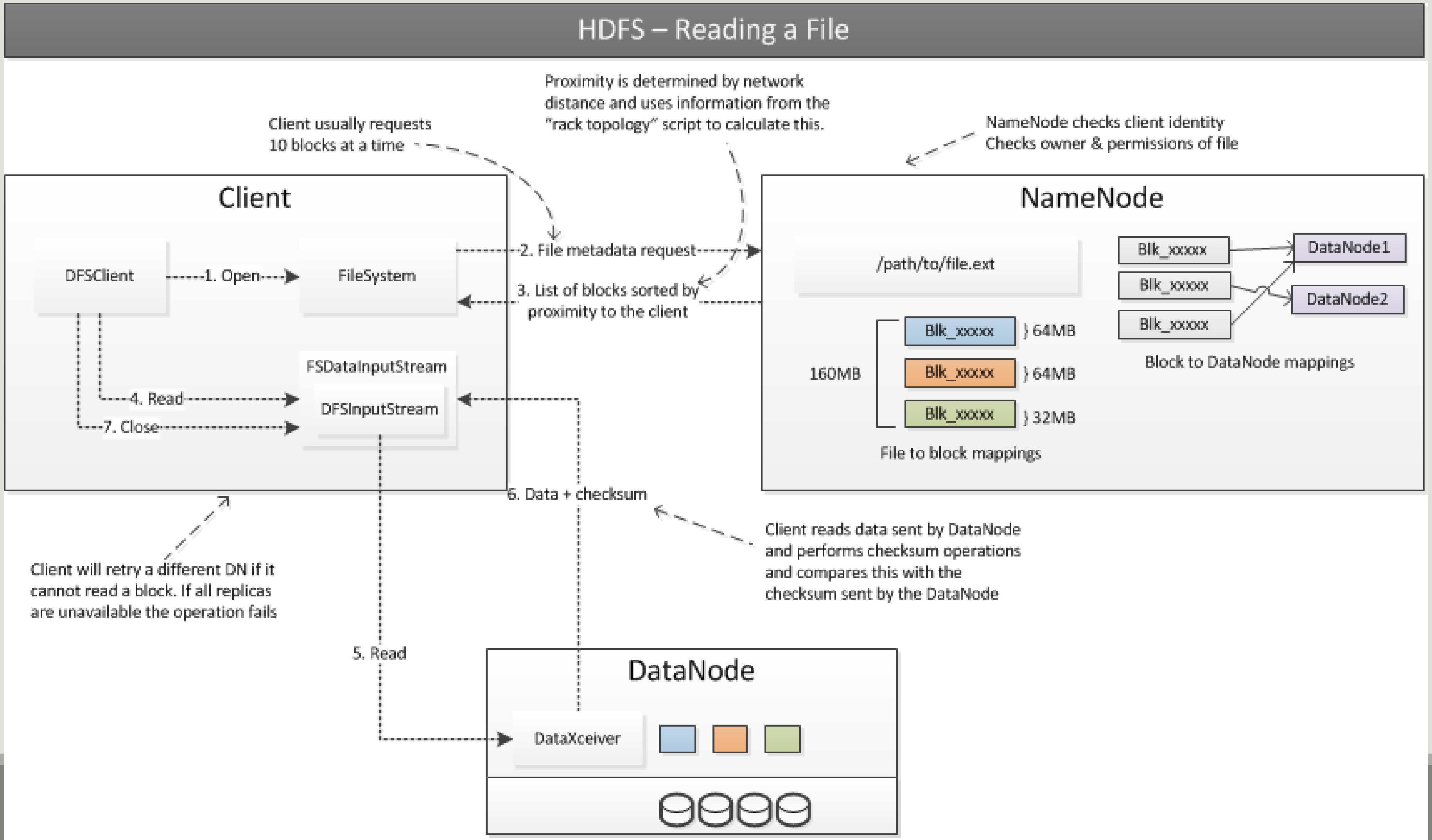
Prior to Hadoop 2, the NN was a single point of failure in an HDFS cluster.

In a typical HA cluster, two separate machines are configured as NNs. At any point in time, exactly one of the NNs is in an *Active* state, and the other is in a *Standby* state.

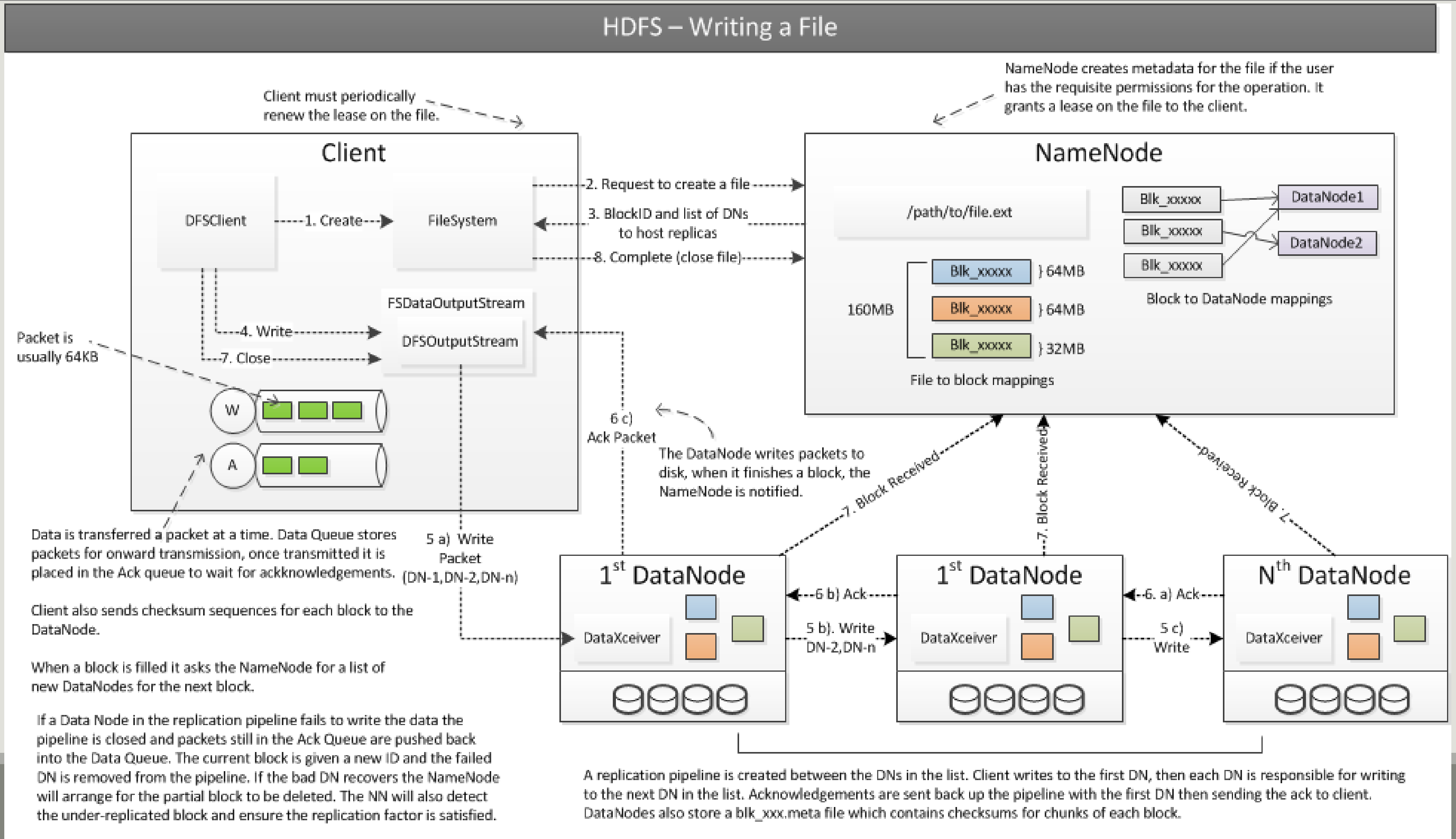
The NN in Standby state keeps informed of the name space meta-data:

- **Reading the log files** written by the active state NN. In the event of a failover, the Standby will ensure that it has read all of the edits from the shared storage before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.
- **Receiving the data block locations** and the heartbeats directly from the DN that are configured with the location of both Name Nodes.

Reading in HDFS


HDFS – Reading a File
Client usually requests 10 blocks at a time
Proximity is determined by network distance and uses information from the "rack topology" script to calculate this.
NameNode checks client identity Checks owner & permissions of file
Client
DFSClient
1. Open
FileSystem
2. File metadata request
3. List of blocks sorted by proximity to the client
FSDataInputStream
DFSInputStream
4. Read
7. Close
NameNode
/path/to/file.ext
Blk_xxxxx
DataNode1
DataNode2
160MB
} 64MB
} 64MB
} 32MB
File to block mappings
Block to DataNode mappings
6. Data + checksum
Client reads data sent by DataNode and performs checksum operations and compares this with the checksum sent by the DataNode
Client will retry a different DN if it cannot read a block. If all replicas are unavailable the operation fails
5. Read
DataNode
DataXceiver

Writing in HDFS



Client must periodically renew the lease on the file.

NameNode creates metadata for the file if the user has the requisite permissions for the operation. It grants a lease on the file to the client.

The DataNode writes packets to disk, when it finishes a block, the NameNode is notified.

Data is transferred a packet at a time. Data Queue stores packets for onward transmission, once transmitted it is placed in the Ack queue to wait for ackknowledgements.

Client also sends checksum sequences for each block to the DataNode.

When a block is filled it asks the NameNode for a list of new DataNodes for the next block.

If a Data Node in the replication pipeline fails to write the data the pipeline is closed and packets still in the Ack Queue are pushed back into the Data Queue. The current block is given a new ID and the failed DN is removed from the pipeline. If the bad DN recovers the NameNode will arrange for the partial block to be deleted. The NN will also detect the under-replicated block and ensure the replication factor is satisfied.

A replication pipeline is created between the DNs in the list. Client writes to the first DN, then each DN is responsible for writing to the next DN in the list. Acknowledgements are sent back up the pipeline with the first DN then sending the ack to client. DataNodes also store a blk_xxx.meta file which contains checksums for chunks of each block.

The Heart Beat Mechanism

Heart Beats are signals DNs periodically (by default 10 minutes) send to the NN to make it aware that they are active.

Heart Beats absence trigger several actions within the cluster:

- If the NN does not receive the Heart Beats from DN it considers it inactive and it creates a replica, on different DNs, of the data blocks stored in such node
- The AM has to emit heartbeats to the RM to keep it informed that the AM is alive and still running.

YARN

YARN is the resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications

While in Hadoop 1 the job scheduling/monitoring functionalities were both taken by the Job tracker, in Hadoop 2 such functionalities are redistributed between the following agents:

- A global **Resource Manager** (RM) that is the ultimate authority that arbitrates resources among all the applications in the system. Resources are negoziated, assigned and used based on the abstract notion of a resource **Container** which incorporates elements such as memory, cpu, disk, network etc. In the first version, only memory is supported.
- A per-node slave, the **Node Manager** (NM) who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the RM.
- A per-application **Application Manager** (AM) is tasked with negotiating resources from the RM and working with the NMs to execute and monitor the tasks.

YARN

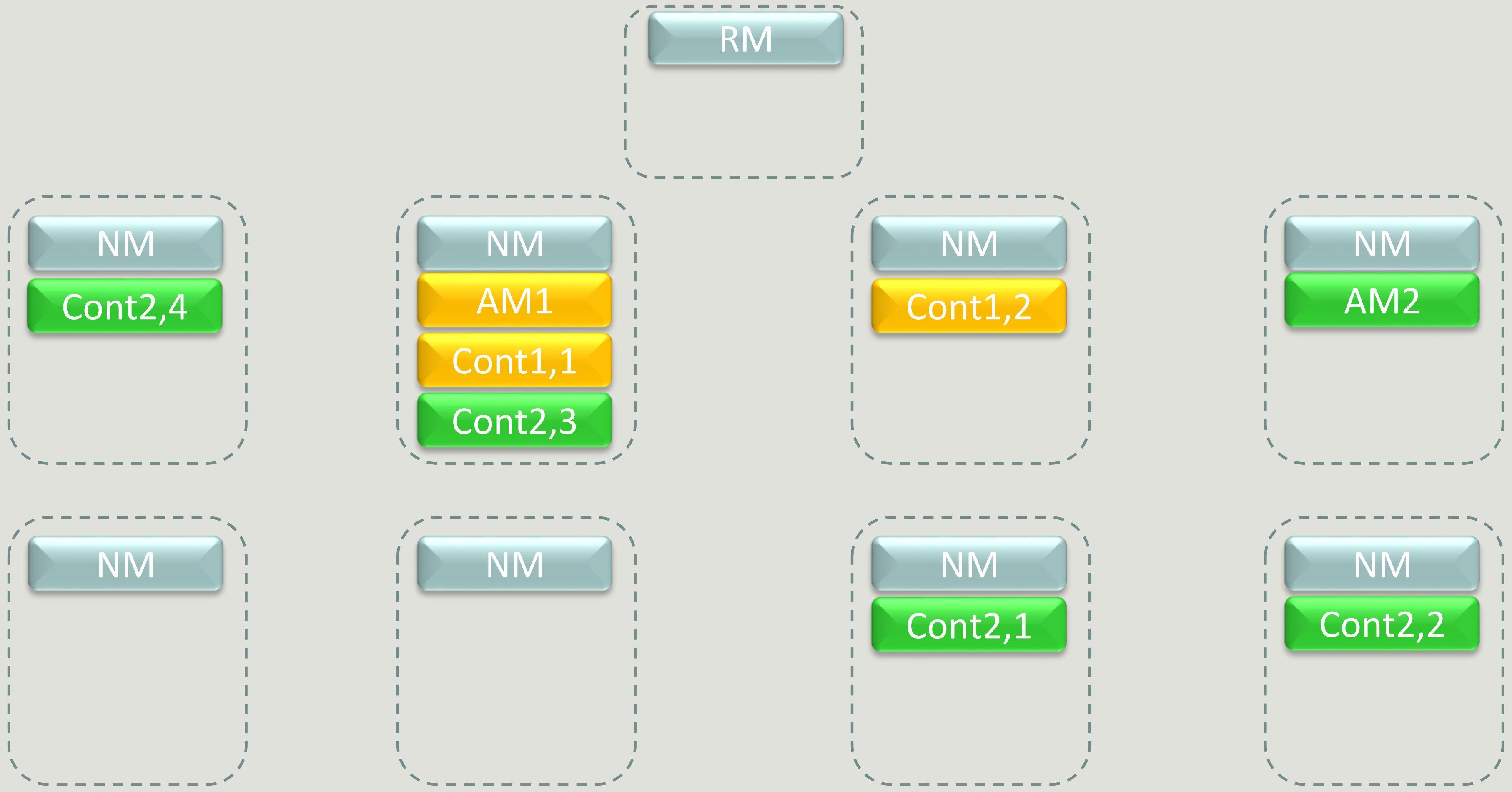
The RM has two main components: Scheduler and Applications Manager.

The **Scheduler** is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc. The Scheduler is a pure scheduler in the sense that:

- it performs no monitoring or tracking of status for the applications.
- it offers no guarantees about restarting failed tasks either due to application failure or hardware failures.

The **ApplicationsManager** (AsM) is responsible for accepting job-submissions, negotiating the first container for executing the application specific AM and provides the service for restarting the AM container on failure.

YARN


RM
NM
Cont2,4
NM
AM1
Cont1,1
Cont2,3
NM
Cont1,2
NM
AM2
NM
NM
NM
Cont2,1
NM
Cont2,2

Application Master

The AM allows YARN to be more:

Scalable: many functionalities are distributed on several AMs thus the RM is no more a bottleneck in the cluster. RM is a *pure scheduler* i.e. it doesn't attempt to provide fault-tolerance for resources.

Open: moving all application framework specific code into the AM generalizes the system so it can now support multiple frameworks such as MapReduce, MPI and Graph Processing.

- Since AM is essentially user-code, RM cannot trust it (i.e. any AM is not a privileged service)
- The YARN system (RM and NM) has to protect itself from faulty or malicious AMs and resources granted to them at all costs

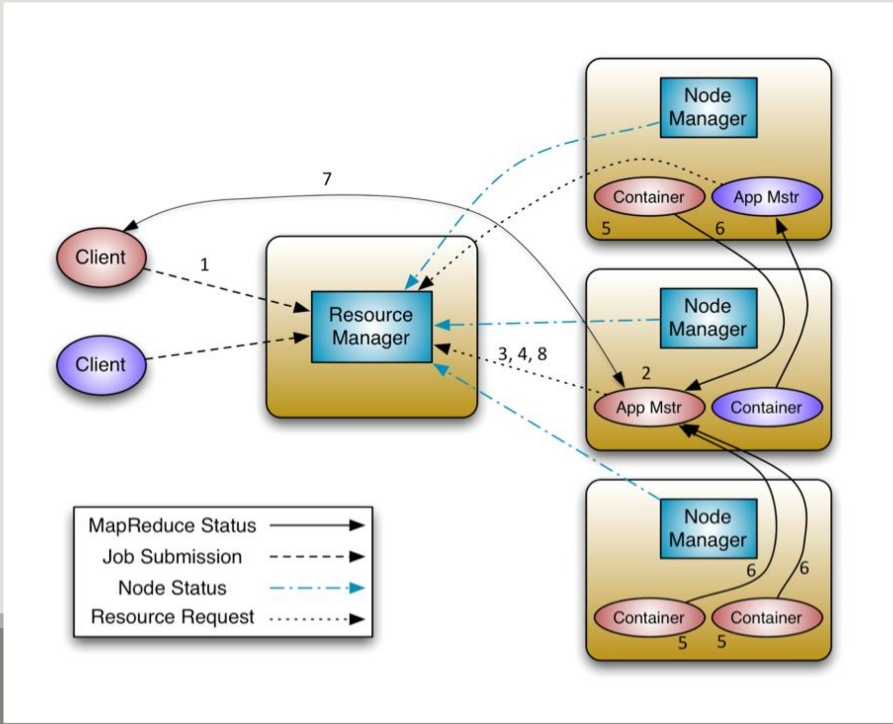
Depending on the Containers it receives from the RM, the AM may update its execution plan to accommodate the excess or lack of resources. Container allocation/de-allocation can take place in a dynamic fashion as the application progresses

YARN – Walkthrough

Application execution consists of the following steps:

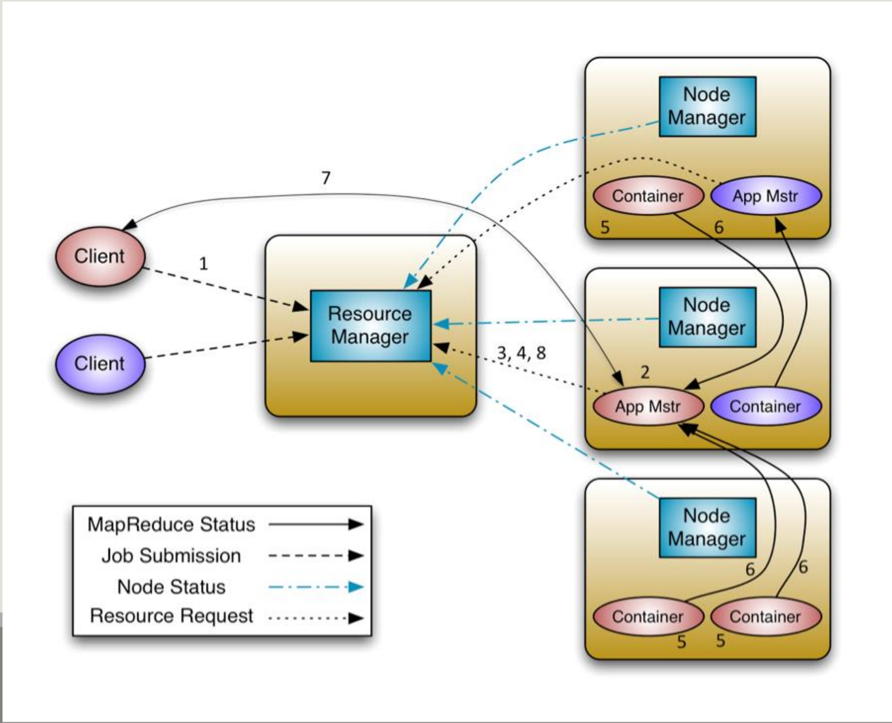
- Application submission
- Bootstrapping the AM instance for the application
- Application execution managed by the AM instance

1. A client program **submits the application**, including the necessary specifications to launch the application-specific AM itself.
2. The RM assumes the responsibility to **negotiate** a specified container in which to start the AM and then **launches** it.
3. The AM, on boot-up, **registers with the RM** – the registration allows the client program to query the RM for details, which allow it to directly communicate with its own AM.



YARN – Walkthrough

4. During normal operation the AM **negotiates** appropriate resource containers via the resource-request protocol.
5. On successful container allocations, the AM **launches** the container by providing the container launch specification to the NM. The launch specification, typically, includes the necessary information to allow the container to communicate with the AM itself.
6. The application code executing within the container then **provides necessary information** (progress, status etc.) to its AM.
7. During the application execution, the client that submitted the program **communicates** directly with the AM to get status, progress updates etc.
8. Once the application is complete, and all necessary work has been finished, the AM **deregisters** with the RM and shuts down, allowing its own container to be repurposed.



Business Intelligence Group