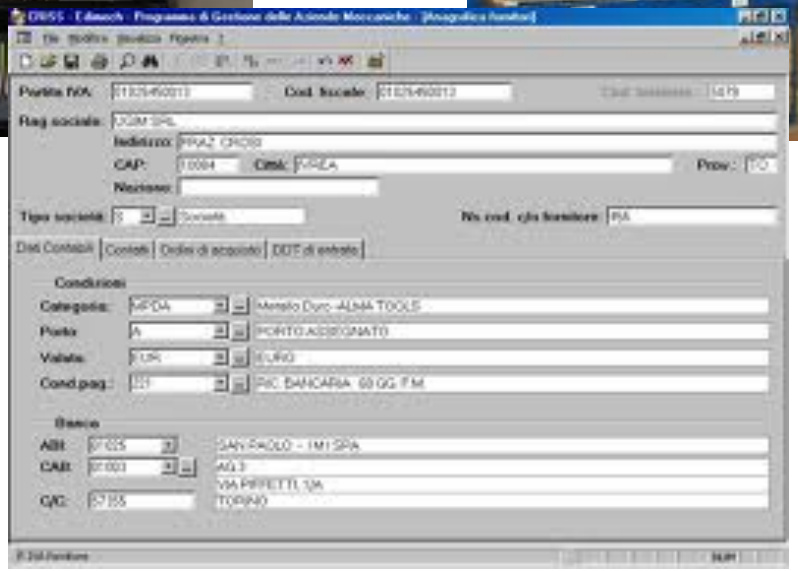
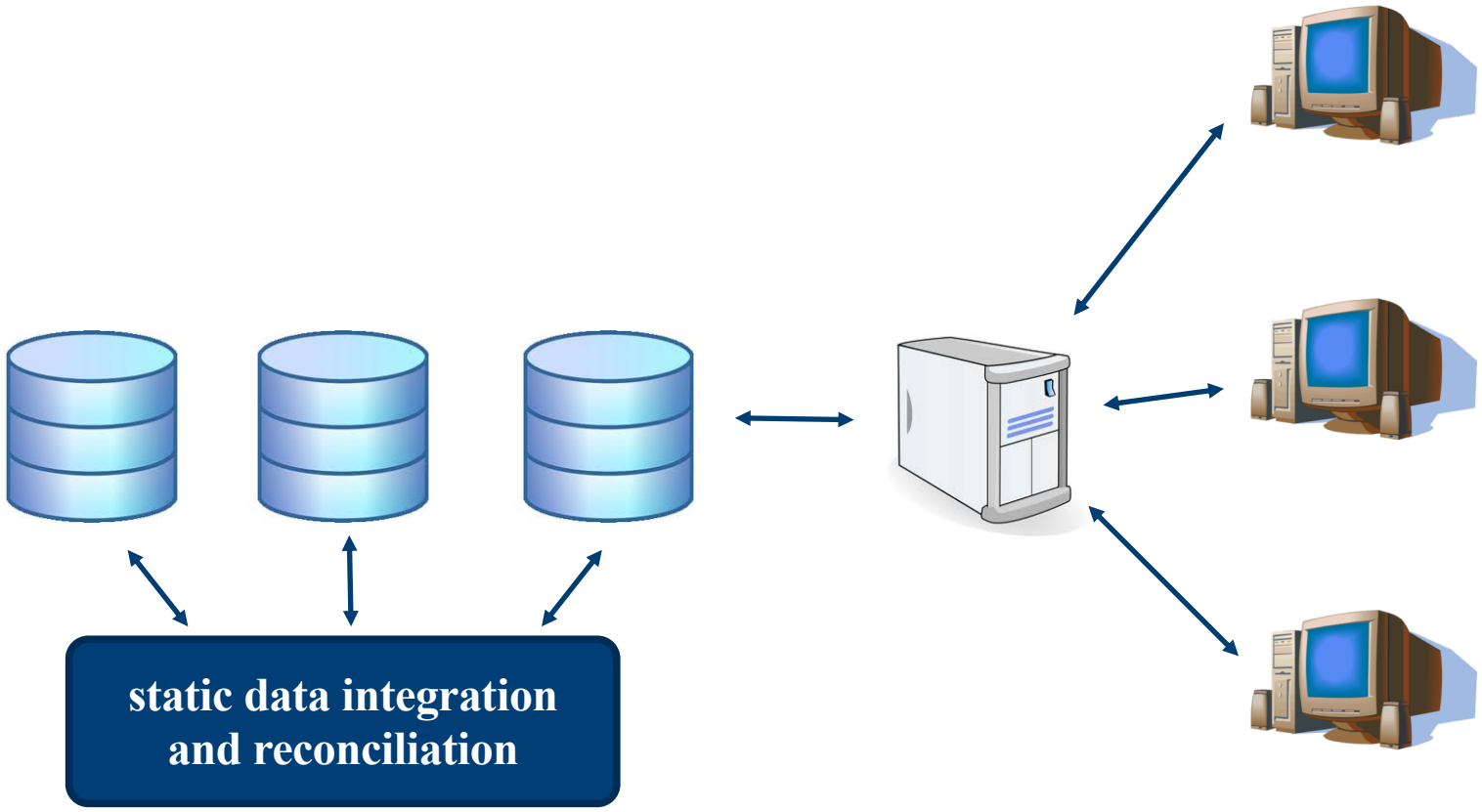
Introduction to Big Data and Large Scale Architectures

What kind of data?

Classical data management applications



Reference architecture



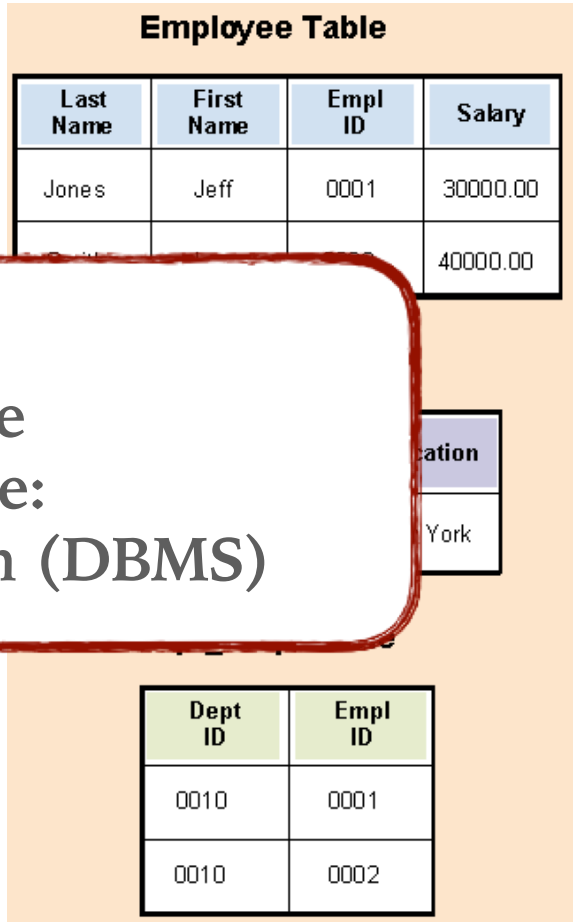
Data models

- Relational model: set of tuples (relations) sharing relationships

- Tu
  sa

- Re
  market

Small enough data
Well-defined structure
A new kind of software:
DataBase Management System (DBMS)



Typical interaction

- Declarative standard language - SQL
- Operational data retrieval operations

Employee Table

Last Name	First Name	Empl ID	Salary
Jones	Jeff	0001	30000.00
Smith	Jane	0002	40000.00

Department Table

Dept ID	Name	Location
0010	Sales	New York

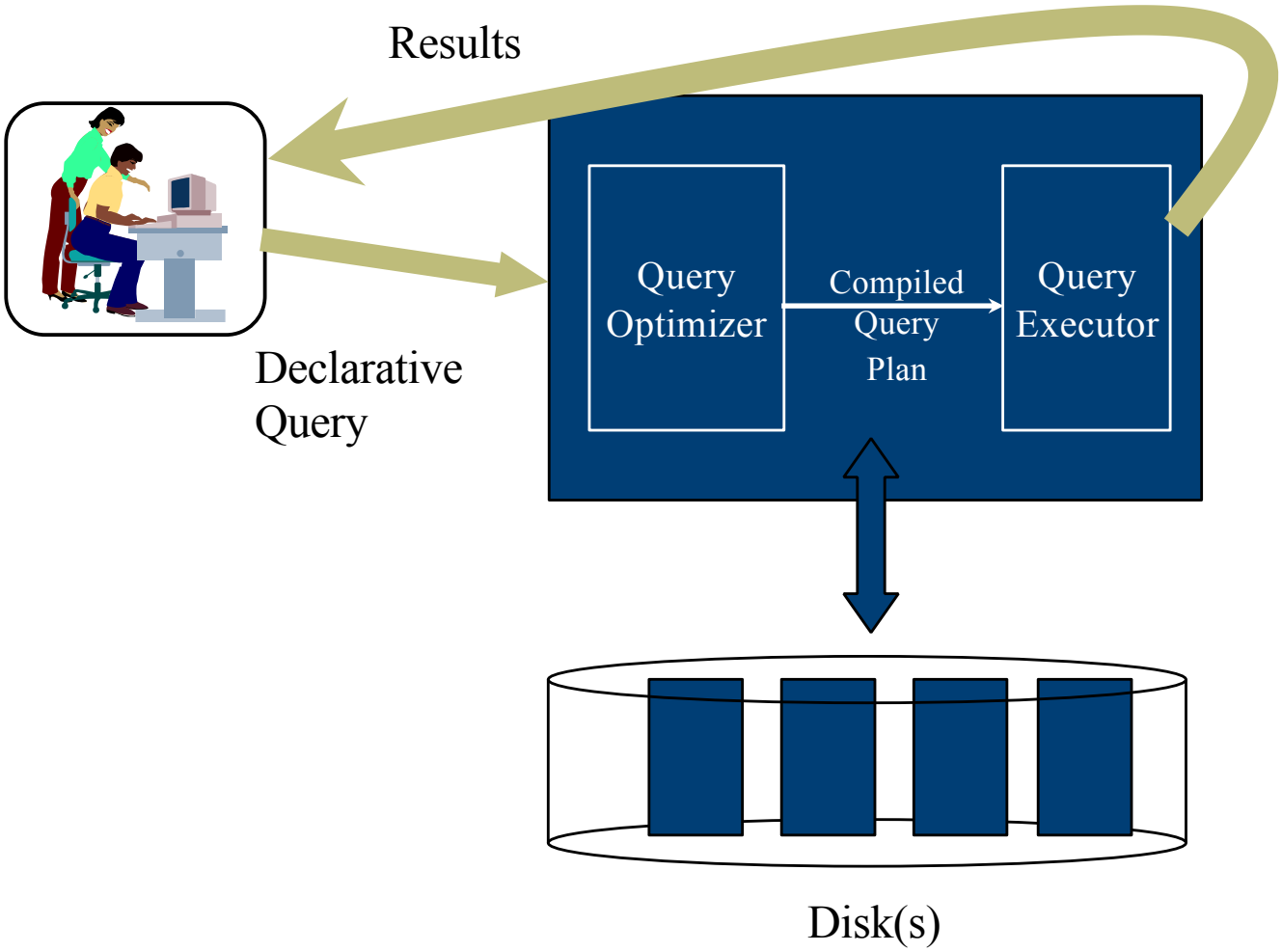
Dept_Empl Table

Dept ID	Empl ID
0010	0001
0010	0002

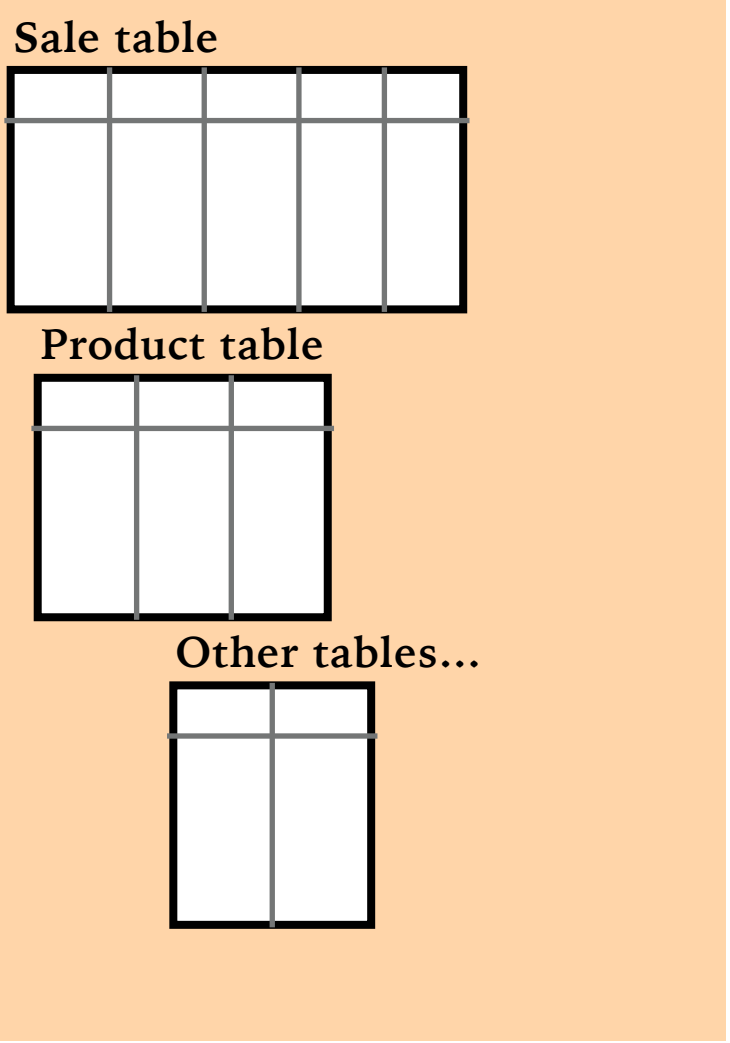
- Selection
  - SELECT * FROM Employee
    WHERE salary > 3000
- Join
  - SELECT *
    FROM Employee NATURAL JOIN Dept_Empl
    NATURAL JOIN Department
    WHERE Location = ‘New York’

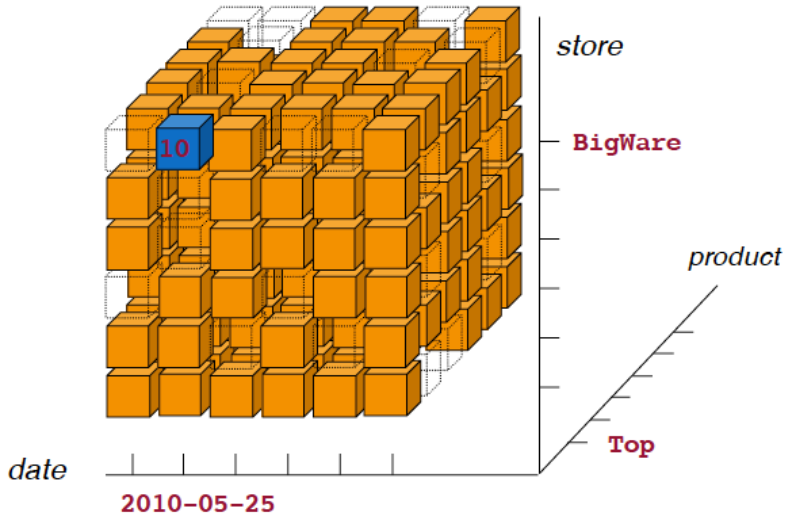
Data management

- efficiency
- integrity
- concurrency
- reliability
- security



Database and data warehouse





transactional vs analytical models and systems

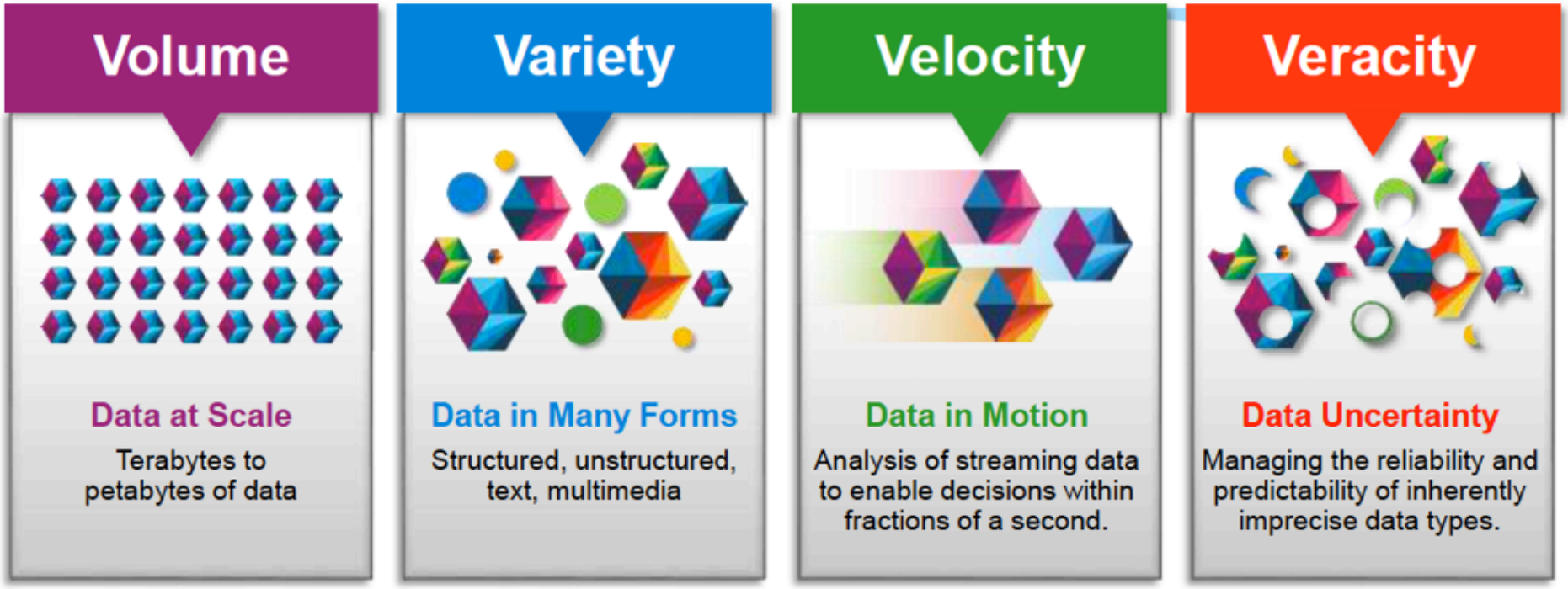
From data management to big data management

"Big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications"

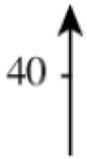
When data become “big”?

IOPS: Input/Output Operations Per Second

The 4 V's of big data

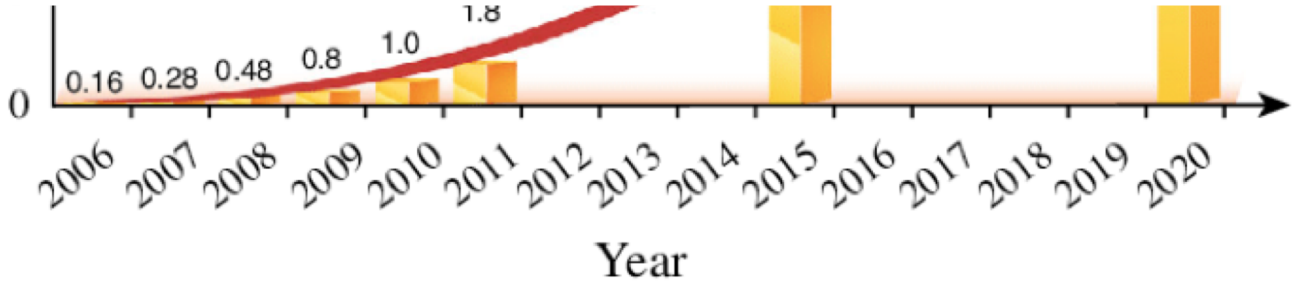


Volume

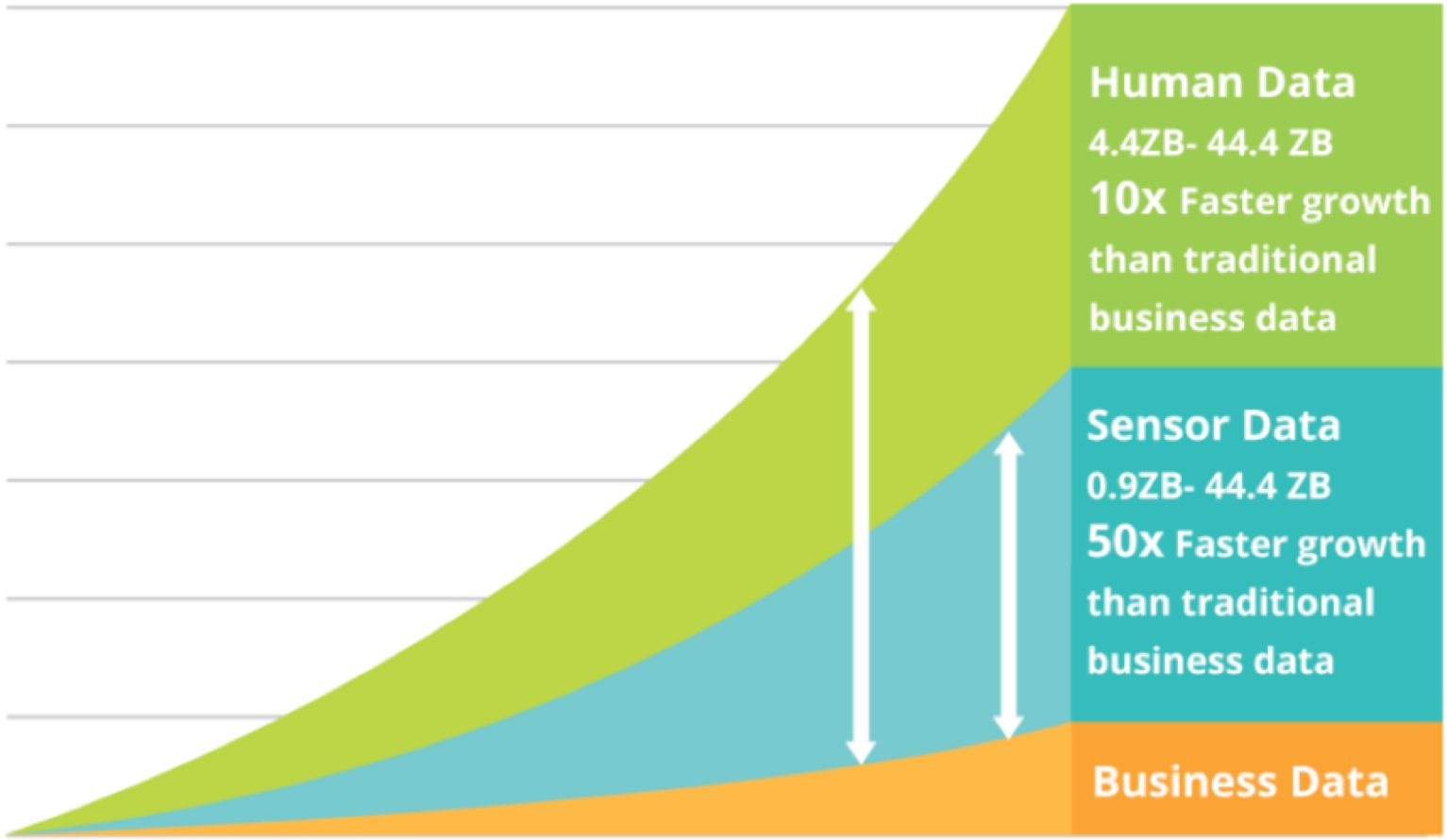


MULTIPLI DEL BYTE

Nome	Simbolo	Multiplo	byte
Kilobyte	kB	10^3	1.000
Megabyte	MB	10^6	1.000.000
Gigabyte	GB	10^9	1.000.000.000
Terabyte	TB	10^{12}	1.000.000.000.000
Petabyte	PB	10^{15}	1.000.000.000.000.000
Exabyte	EB	10^{18}	1.000.000.000.000.000.000
Zettabyte	ZB	10^{21}	1.000.000.000.000.000.000.000
Yottabyte	YB	10^{24}	1.000.000.000.000.000.000.000.000

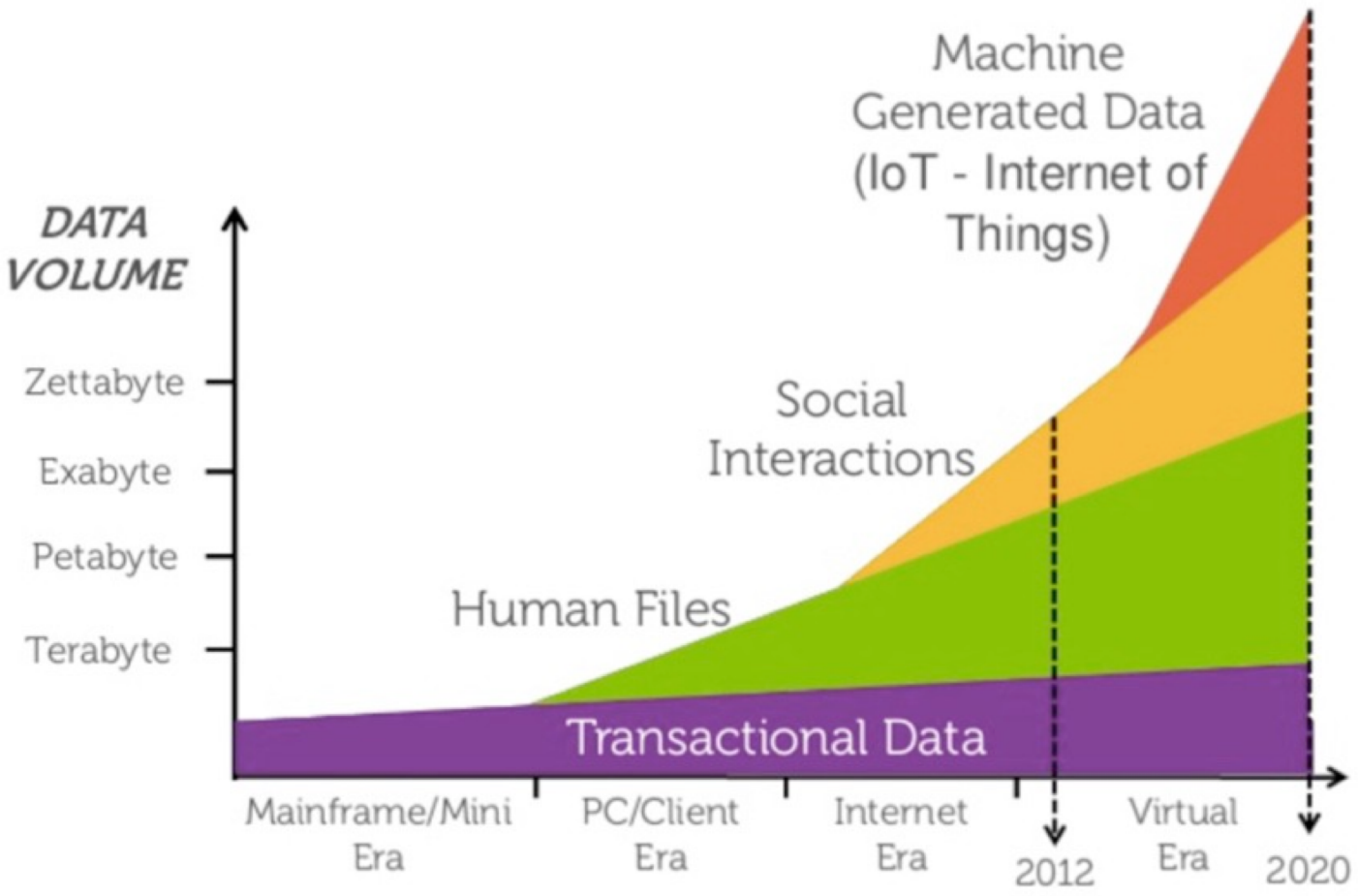


Volume

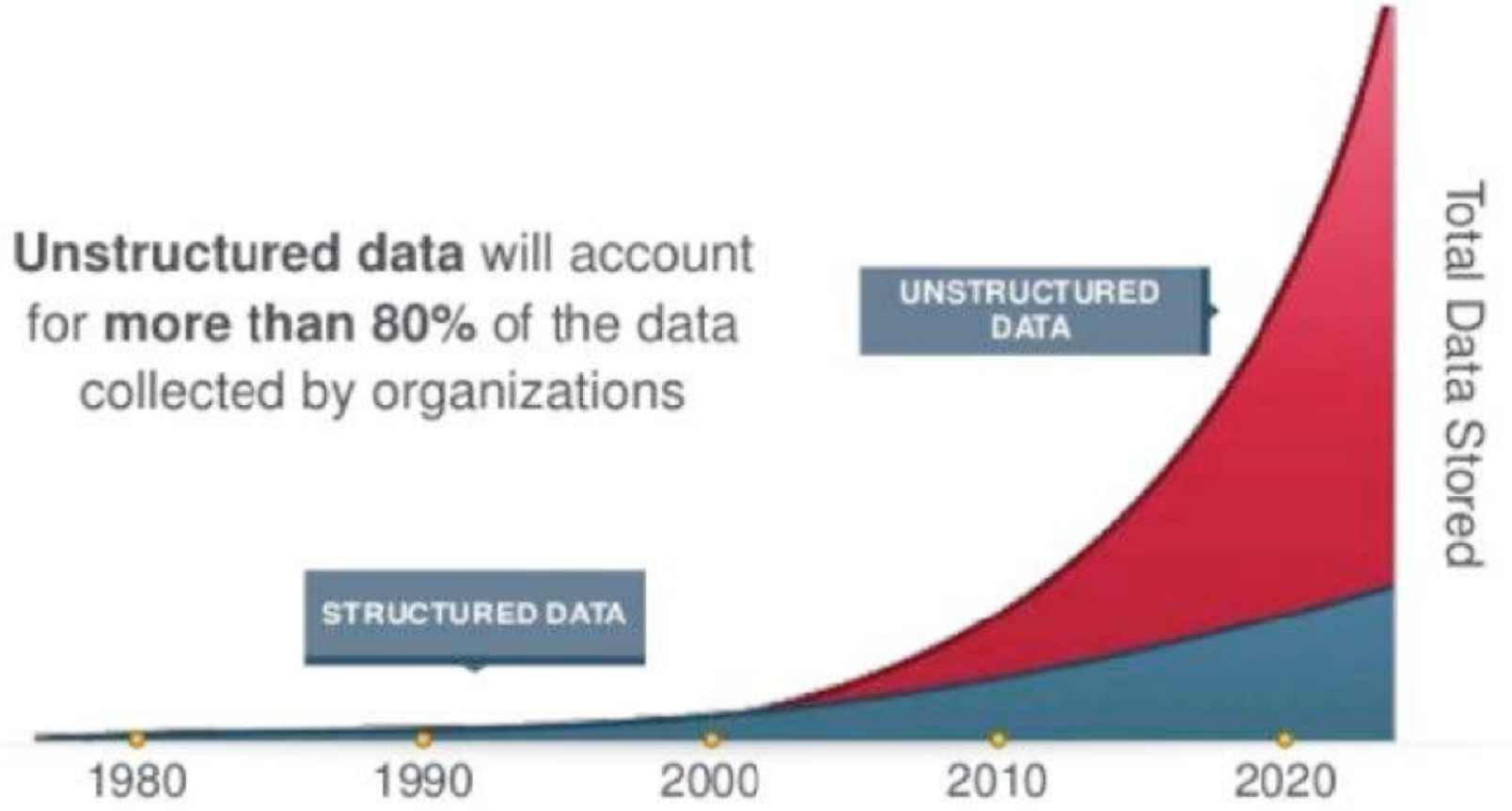


Source: Inside big data

Variety

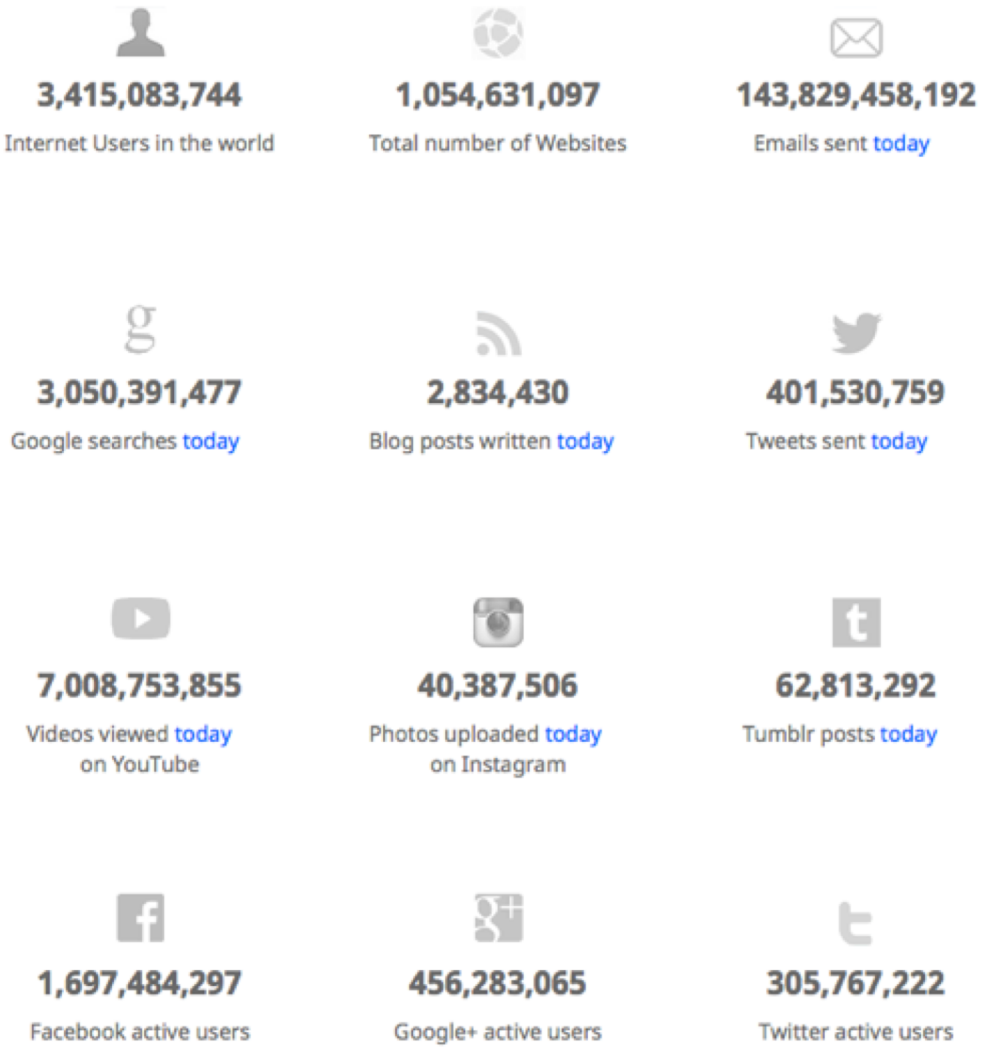

Machine Generated Data (IoT - Internet of Things)
DATA VOLUME
Zettabyte
Exabyte
Petabyte
Terabyte
Social Interactions
Human Files
Transactional Data
Mainframe/Mini Era
PC/Client Era
Internet Era
2012
Virtual Era
2020

Variety

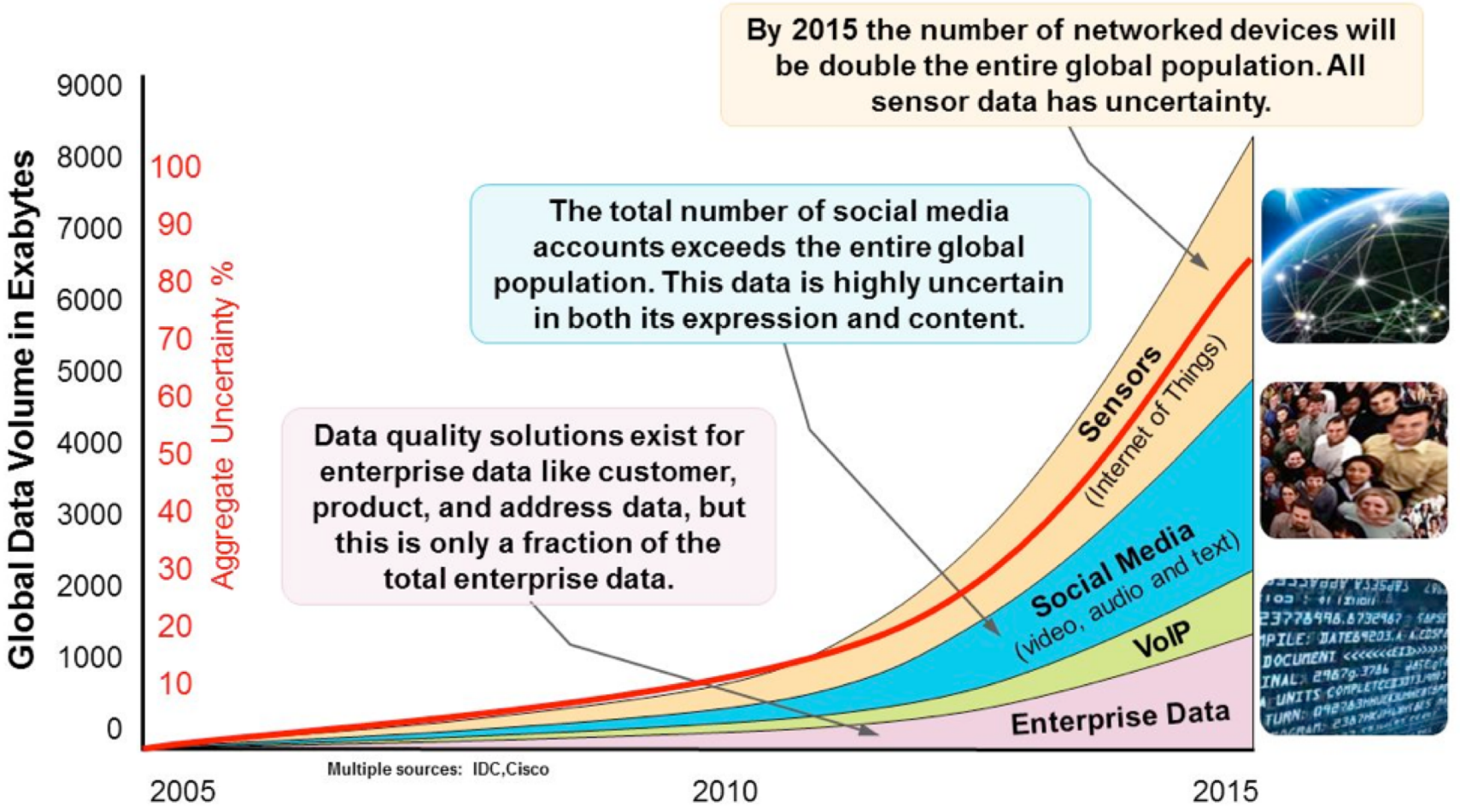

Unstructured data will account for **more than 80%** of the data collected by organizations
UNSTRUCTURED DATA
STRUCTURED DATA
Total Data Stored
1980
1990
2000
2010
2020

Velocity

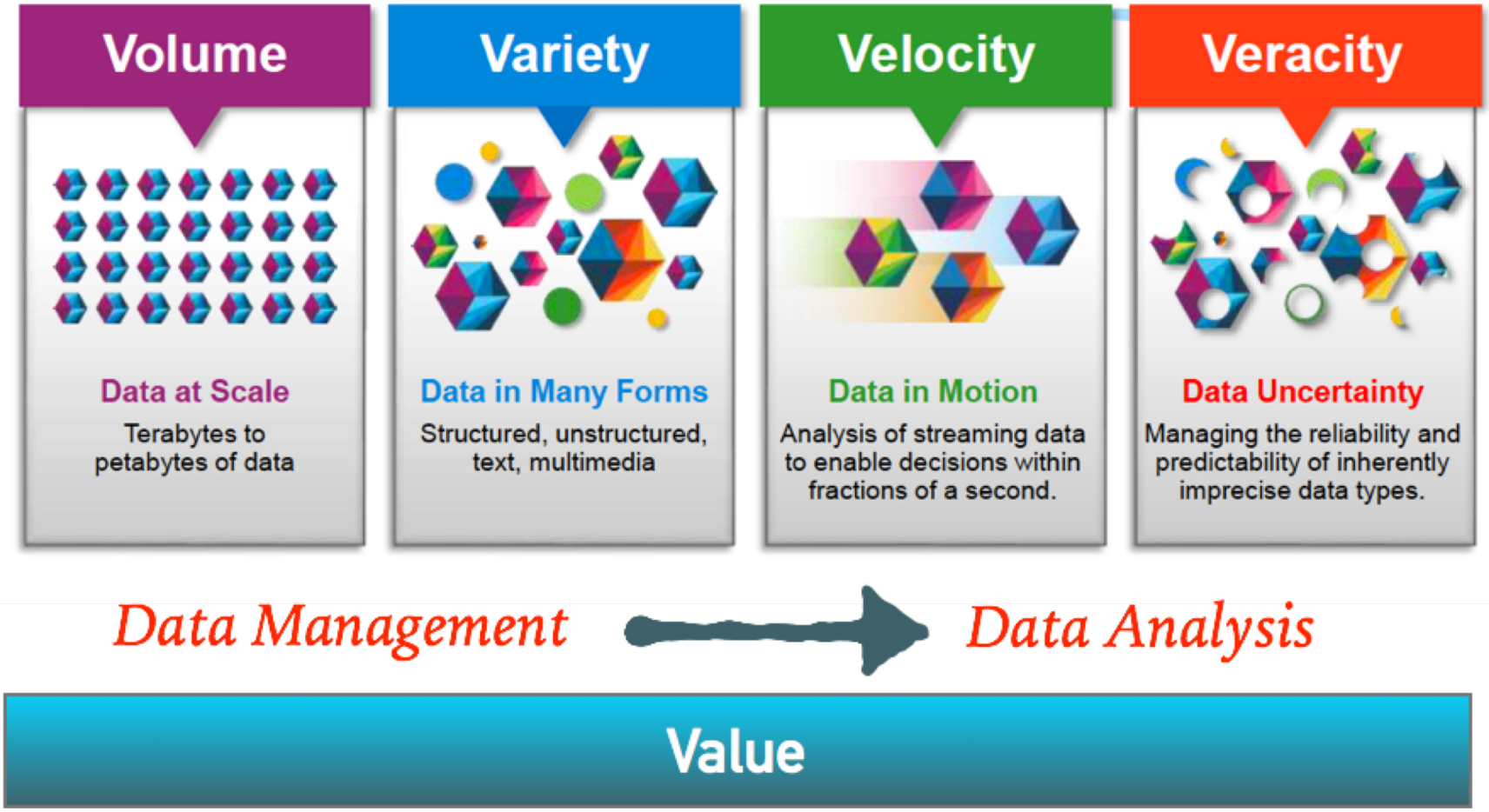
www.internetlivestats.com/



Veracity


By 2015 the number of networked devices will be double the entire global population. All sensor data has uncertainty.
The total number of social media accounts exceeds the entire global population. This data is highly uncertain in both its expression and content.
Data quality solutions exist for enterprise data like customer, product, and address data, but this is only a fraction of the total enterprise data.
Global Data Volume in Exabytes
Aggregate Uncertainty %
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
100
90
80
70
60
50
40
30
20
10
Sensors
(Internet of Things)
Social Media
(video, audio and text)
VoIP
Enterprise Data
Multiple sources: IDC,Cisco
2005
2010
2015

From 4 to 5 V's


Volume
Data at Scale
Terabytes to petabytes of data
Variety
Data in Many Forms
Structured, unstructured, text, multimedia
Velocity
Data in Motion
Analysis of streaming data to enable decisions within fractions of a second.
Veracity
Data Uncertainty
Managing the reliability and predictability of inherently imprecise data types.
Data Management
Data Analysis
Value

The real need

… need special hardware, algorithms, tools to work at this scale

… need distribution at this scale

Distributed systems in short

- A distributed system is an application that coordinates the actions of several computers to achieve a specific task
- This coordination is achieved by exchanging messages which are pieces of data that convey some information
- The system relies on a network that connects the computers and handles the routing of messages

Distributed systems in short

- We distinguish between
  - server node: it provides a service of the distributed system
  - client node: it consumes this service
- Nothing prevents a client node to run on the same computer than a server node (this is typically the case in P2P networks)
- Both Client (nodes) and Server (nodes) are communicating software components: we assimilate them with the machines they run on

Distributed data system

Any distributed system that provide commonly needed functionalities for storing, accessing, and processing data, *by distributing data, load and, sometimes, control over the network*

Distributed data systems are standard building blocks for developing data-intensive applications

Compute- vs Data-intensive computing/applications

- Compute-intensive
  - computing applications which devote most of their execution time to computational requirements
  - CPU cycles are the bottleneck

- Data-intensive
  - computing applications which require large volumes of data and devote most of their processing time to I/O and data manipulation
  - The quantity and the complexity of data as well as the speed at which they are changing are the bottleneck

Data intensive application

- The development of data-intensive applications rely on distributed data systems

- Such distributed data systems are large-scale
  - High volume of data
  - High number of nodes

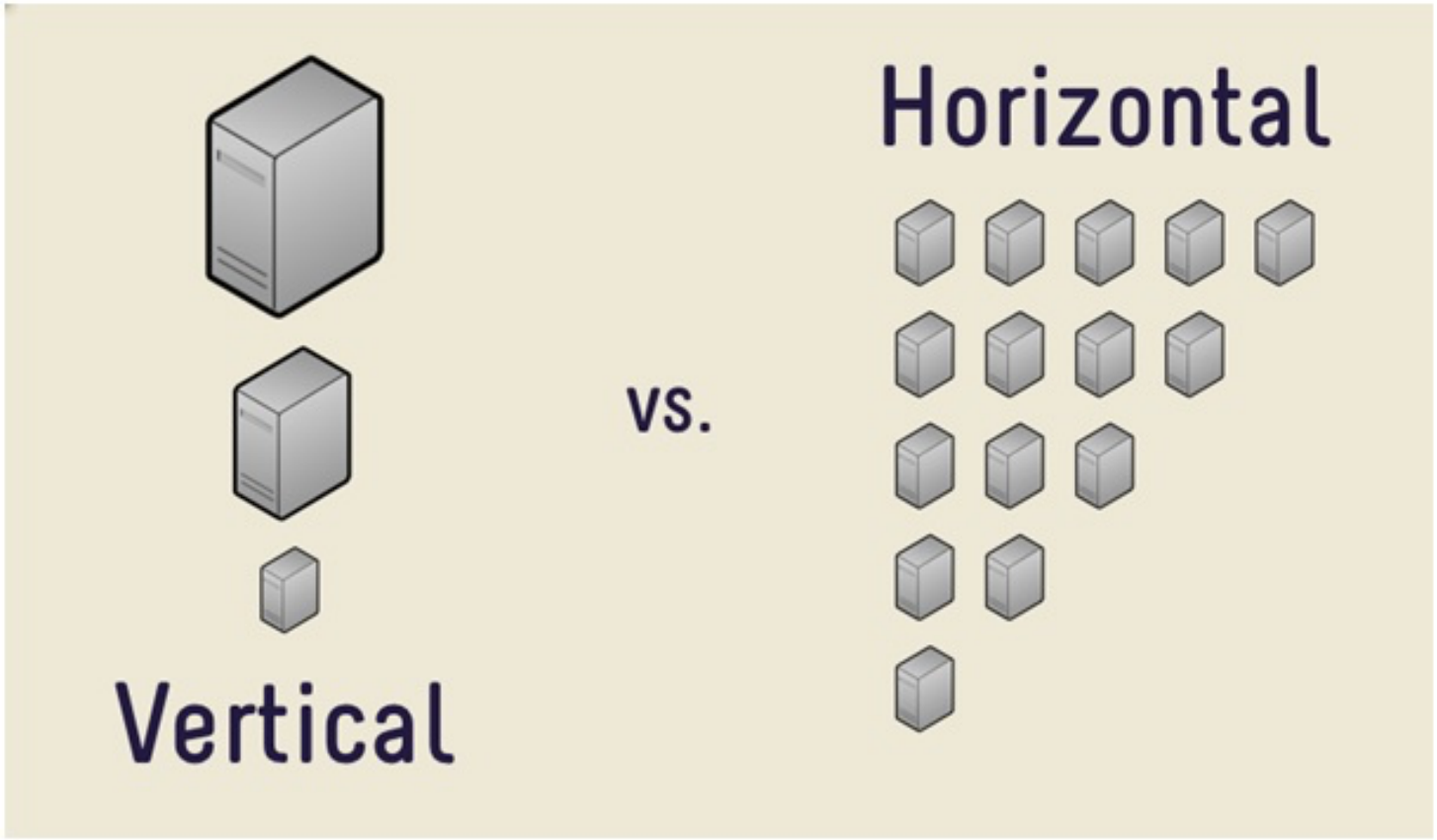
Large Scale Architectures

For Computer Science – Data Science & Engineering students:
this part has been taken from the material of the «Large Scale Computing» course

Computing at scale

- Big data applications require huge amounts of processing and data
  - Measured in petabytes, millions of users, billions of objects
  - Need special hardware, algorithms, tools to work at this scale
- Scaling is an issue

Vertical scaling vs horizontal scaling



What if one computer is not enough?

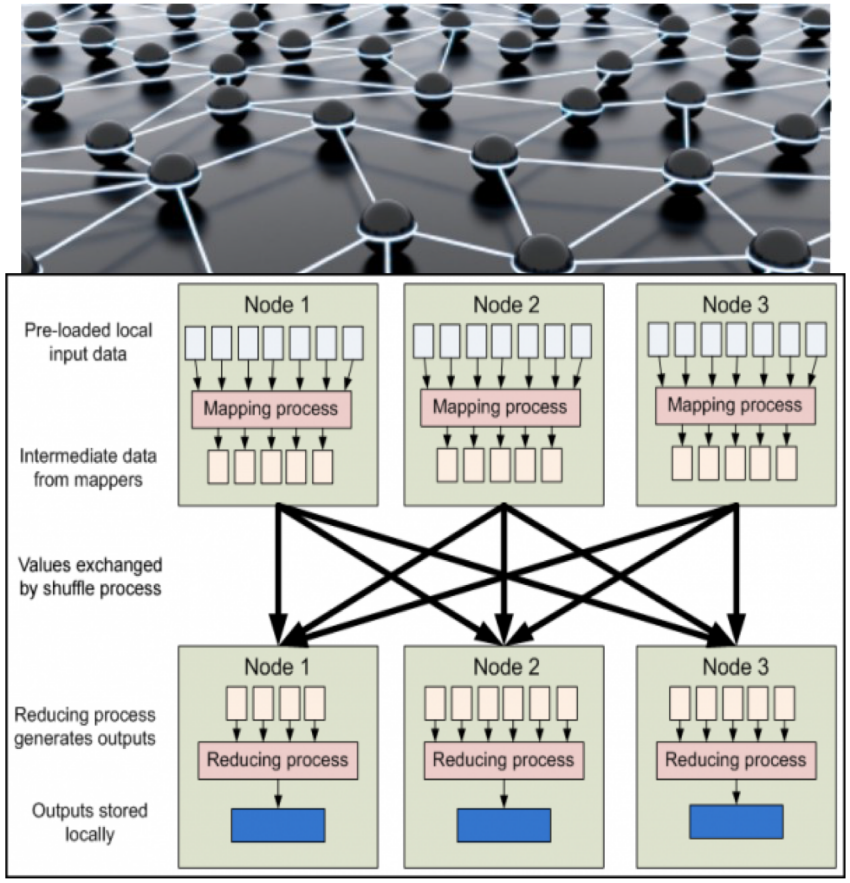
Buy a bigger (server-class) computer – Vertical scaling

What if the biggest computer is not enough?

Buy many computers (cluster) – horizontal scaling

Key ingredients for computing at scale

- *Networking infrastructure*
  - Clusters of computers that work together to a common goal can provide the needed resources
- *Distributed data*
  - For parallel processing
- *Distributed processing*
  - Shared-nothing model
  - New parallel programming paradigms

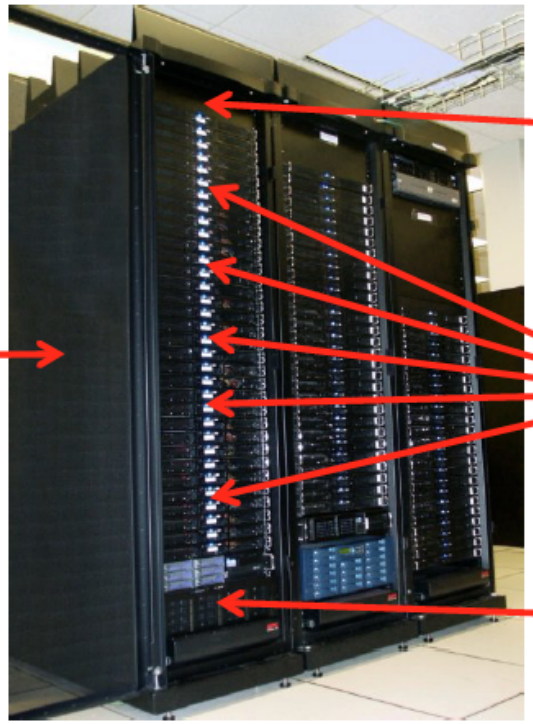


Computer clusters

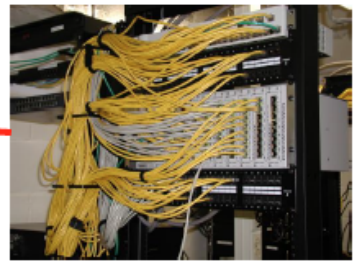
Set of loosely or tightly connected computers that work together so that, in many aspects, they can be viewed as a single system

- Rely on shared-nothing architecture designed for data intensive computing
- Similar, very simple, basic and cheap components, available in large numbers
- Close interconnection (same room?)
- Each node set to perform the same tasks
- A cluster can be owned and used by a single organization or be available on the cloud

Cluster architecture



Rack



Network switch
(connects nodes with each other and with other racks)

Many nodes/blades
(often identical)

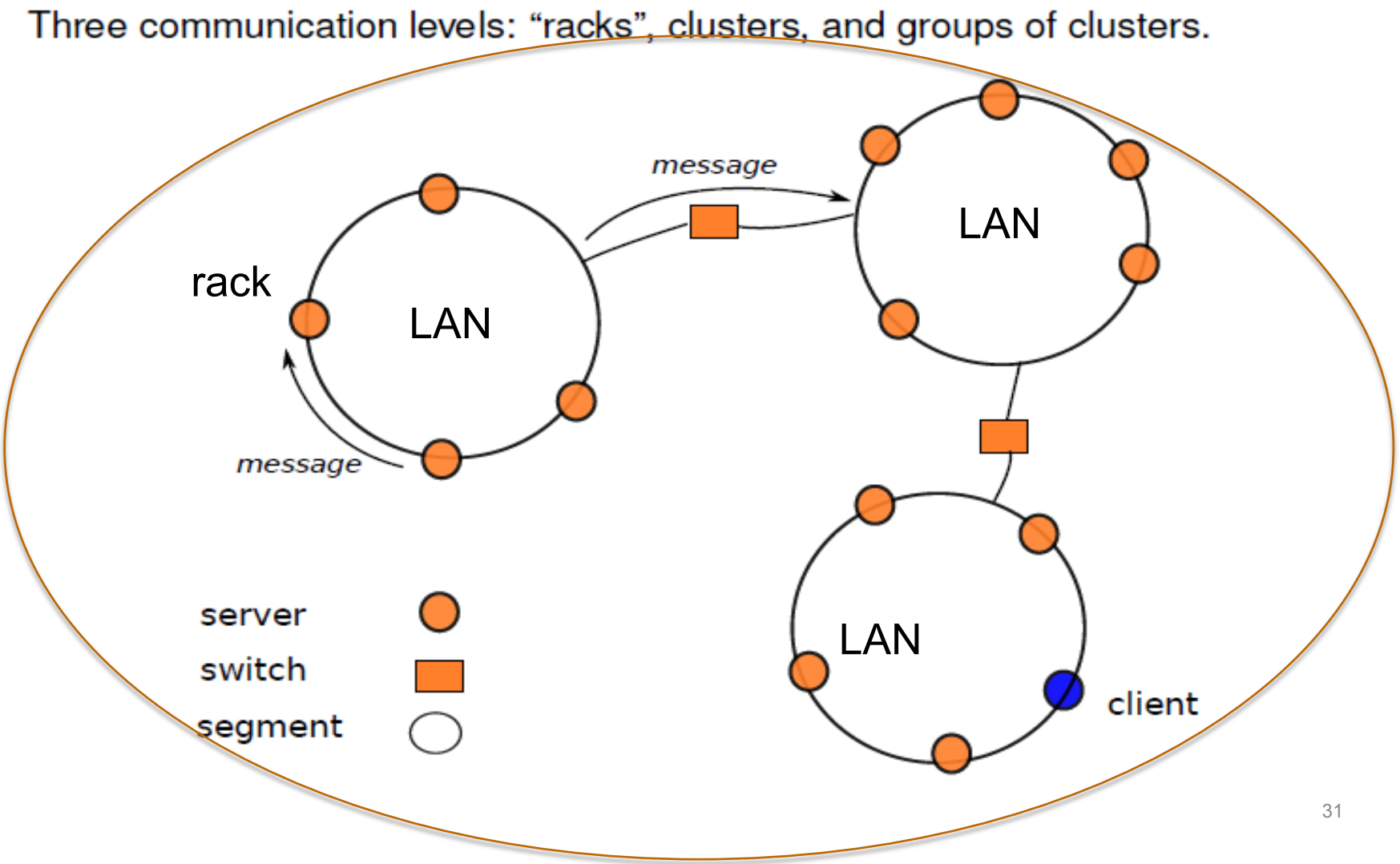
Storage device(s)

Computer nodes (servers) are stored on racks

- 8–64 compute nodes on a rack

Cluster architecture

Capacity – bandwidth: the amount of information that can be transmitted over the channel in a given time unit



Power and cooling

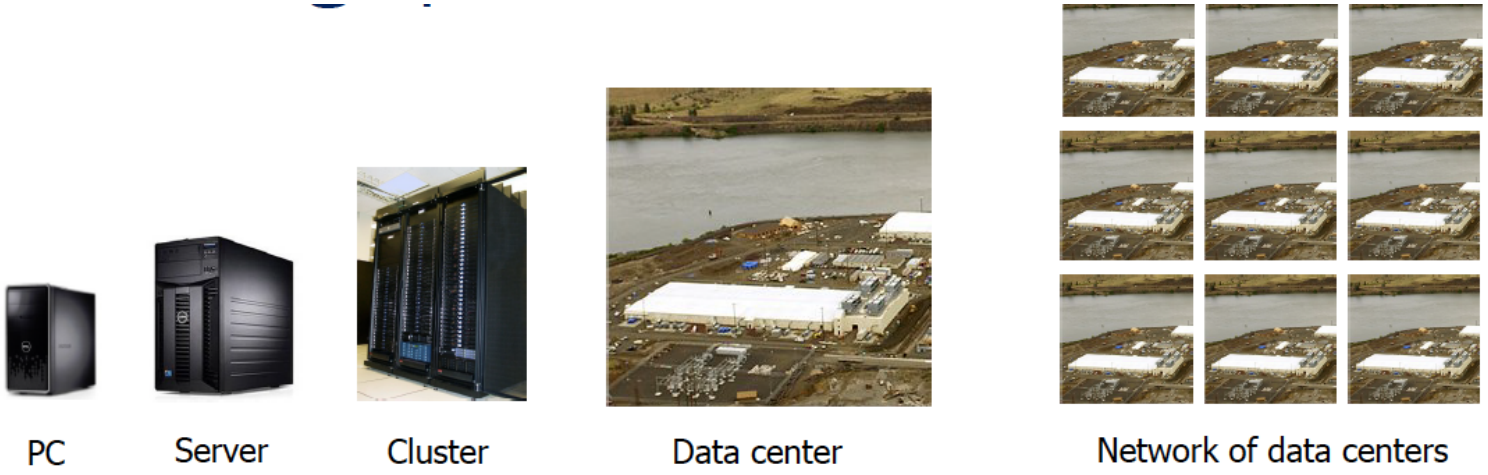
- Clusters need lots of power
  - Example: 140 Watts per server
  - Rack with 32 servers: 4.5kW (needs special power supply!)
  - Most of this power is converted into heat
- Large clusters need massive cooling

Further scaling out (1)

PC Server Cluster Data center

- What if your cluster is too big (hot, power hungry) to fit into your office building?
  - Build a separate building for the cluster
  - Building can have lots of cooling and power
  - Result: data center (= a huge cluster, with additional special hardware)
- Hundreds or thousands of racks
- It can be owned by a single organization and used by several organizations (often according to cloud computing principles)

Further scaling out (2)



- What if even a data center is not big enough?
  - Build additional data centers
- Data centers are often globally distributed

Example: google data centers

Typical setting of a Google data center.

1. $\approx$ 40 servers per rack;
2. $\approx$ 150 racks per data center (cluster);
3. $\approx$ 6,000 servers per data center;
4. how many clusters? Google's secret, and constantly evolving ...

Rough estimate: 150-200 data centers? 1,000,000 servers?

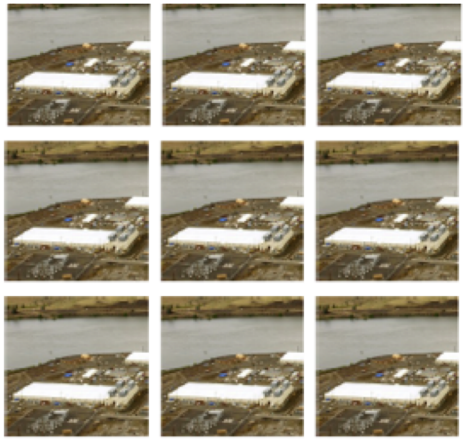
Back to scaling

VERTICAL SCALING

Server

Cluster

Data center



Network of data centers

HORIZONTAL SCALING

PC

ALL SHARED NOTHING ARCHITECTURES

Architectural models

What is the best architecture for developing data-intensive applications?

Some numbers

Exchanging data through the Internet is slow and unreliable with respect to LAN's

Latency: time to initiate an operation
Bandwidth: amount of information transmitted over a channel in a given time unit

Type	Latency	Bandwidth
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.);	At best 100 MB/s
LAN	$\approx 1 - 2 \times 10^{-3}$s (1-2 millisec.);	$\approx$ 1GB/s (single rack); $\approx$ 100MB/s (switched);
Internet	Highly variable. Typ. 10-100 ms.;	Highly variable. Typ. a few MBs.;

1. It is approx. one order of magnitude faster to exchange in-memory data between 2 computers machines in the same rack than for a single computer to read the same data written on the disk
2. But bandwidth is a shared resource…
3. Similar situation for latency, but less impressive

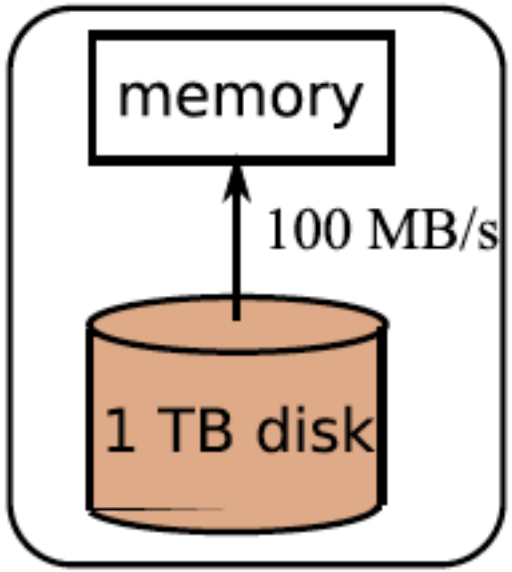
Reference scenario – analytical application

- Sequentially read a large dataset from disks
  - batch operations on the whole collection
  - particularly relevant for applications where most files are written once (by appending new content), then read many times

- Under this assumption:
  - the seek time (time to position the head on the appropriate disk track) is negligible regarding the transfer time
  - the advantages of data distribution (partitioning) and task parallelization become clearer

Why data distribution?

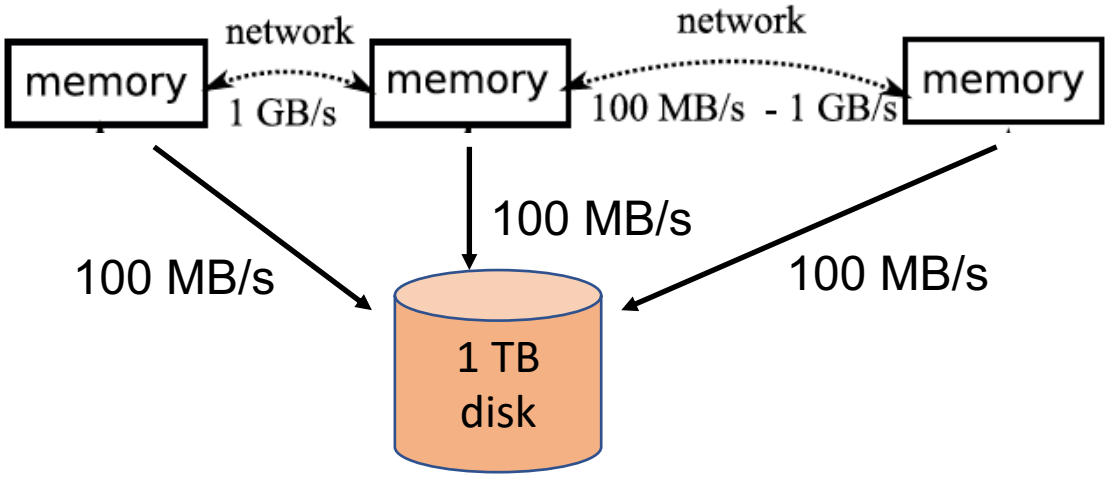
- Centralized database
- No data distribution, sequential access
  - It takes 166 minutes (more than 2 hours and a half) to read 1 TB from disk



a. Single CPU, single disk

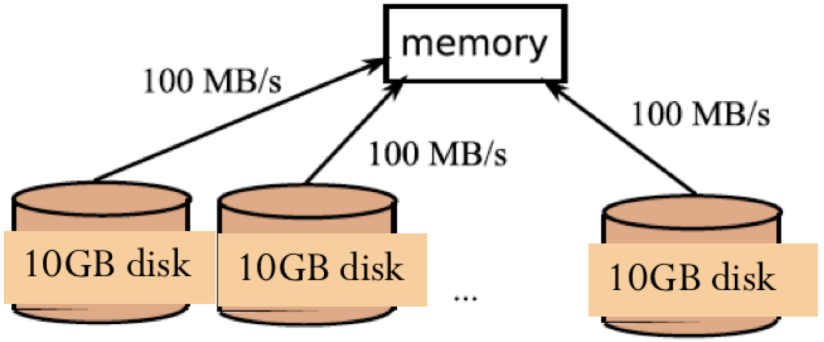
Why data distribution?

- **Shared disk architecture**
- No data distribution, distributed access and processing
- With 100 computers but a shared disk, this works as long as the task is CPU intensive (as in High Performance Computing), but becomes unsuited if large data exchanges are involved



Why data distribution?

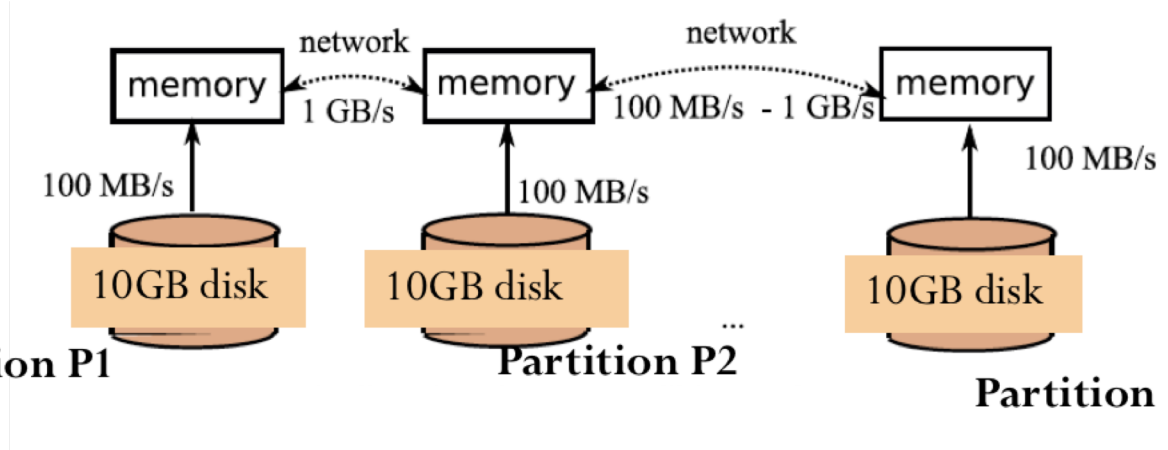
- Shared memory and processor architecture
- Data distribution, shared processor, parallel access
  - With 100 disks, assuming the disks work in parallel and sequentially (about 10 GBs from each disk): about 1mn 30s
  - Data partitioning
    - Splitting a big database into smaller subsets called partitions so that different partitions can be assigned to different nodes (also known as *sharding*)
  - When the size of the data set increases, the CPU of the computer is typically overwhelmed at some point by the data flow and it is slowed down



b. Parallel read: single CPU, many disks

Why data distribution?

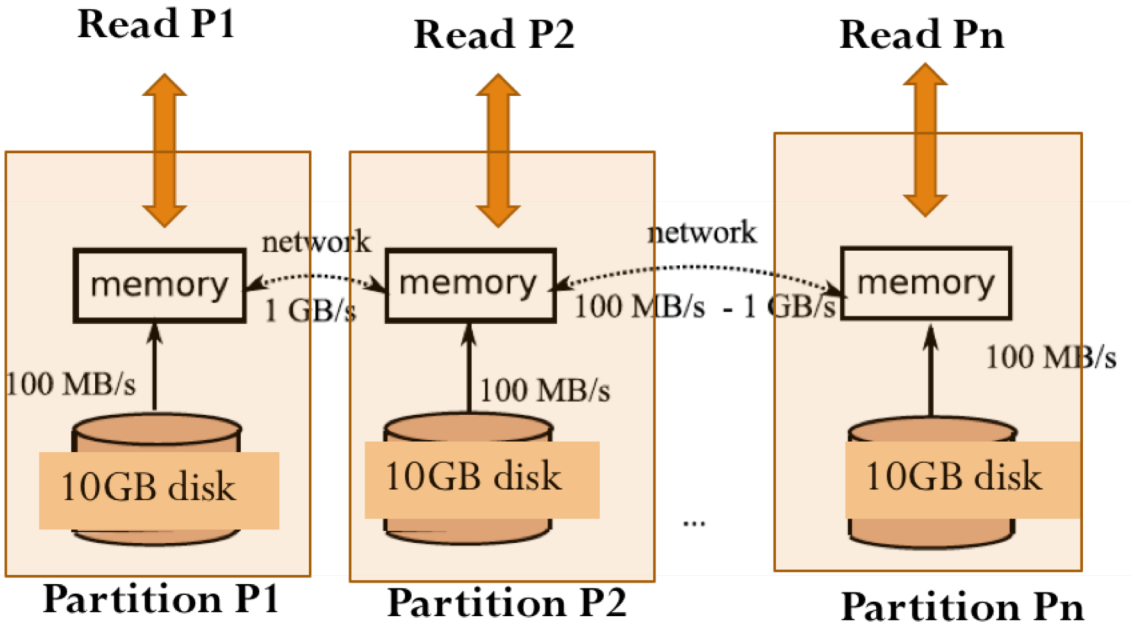
- Shared nothing architecture
- Distributed data, distributed access
  - With a cluster of 100 computers, each disposing of its own local disk: each CPU processes its own dataset
- Data partitioning



This solution is scalable, by adding new computing resources

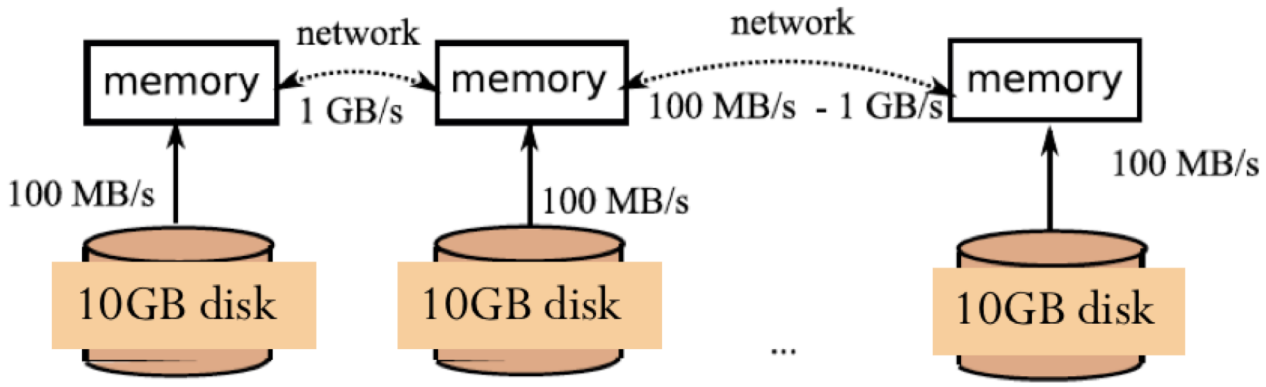
Side effect of data partitioning: task parallelization

- **Shared nothing architecture**
- Distributed data, distributed access: pushing the program near to the data
- Task parallelization (if the task can be parallelized!): the same operation executed over distinct partitions, by different nodes



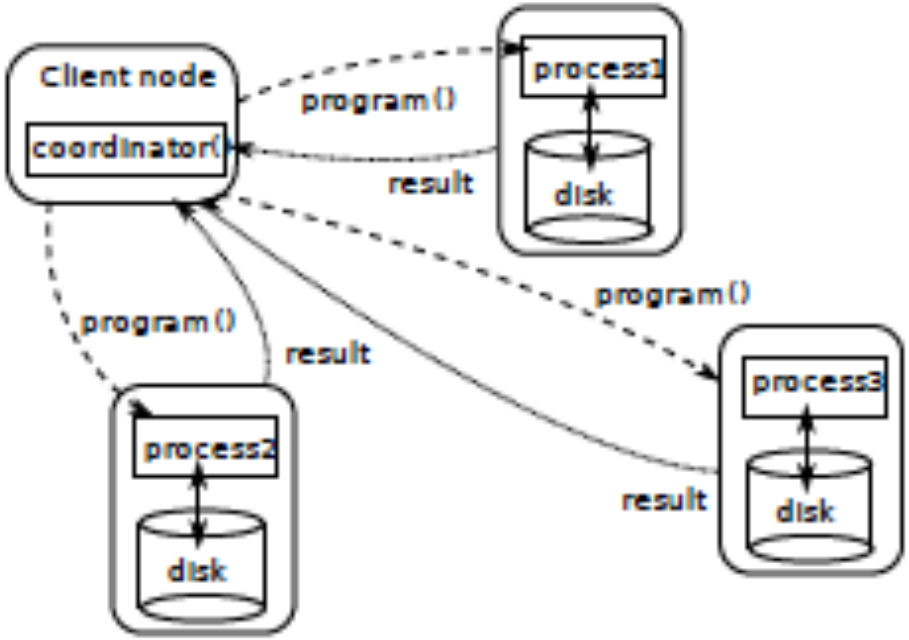
What about control distribution?

- P2P architecture
- Control is distributed



What about control distribution?

- One Master - Many Slaves
- Centralized control

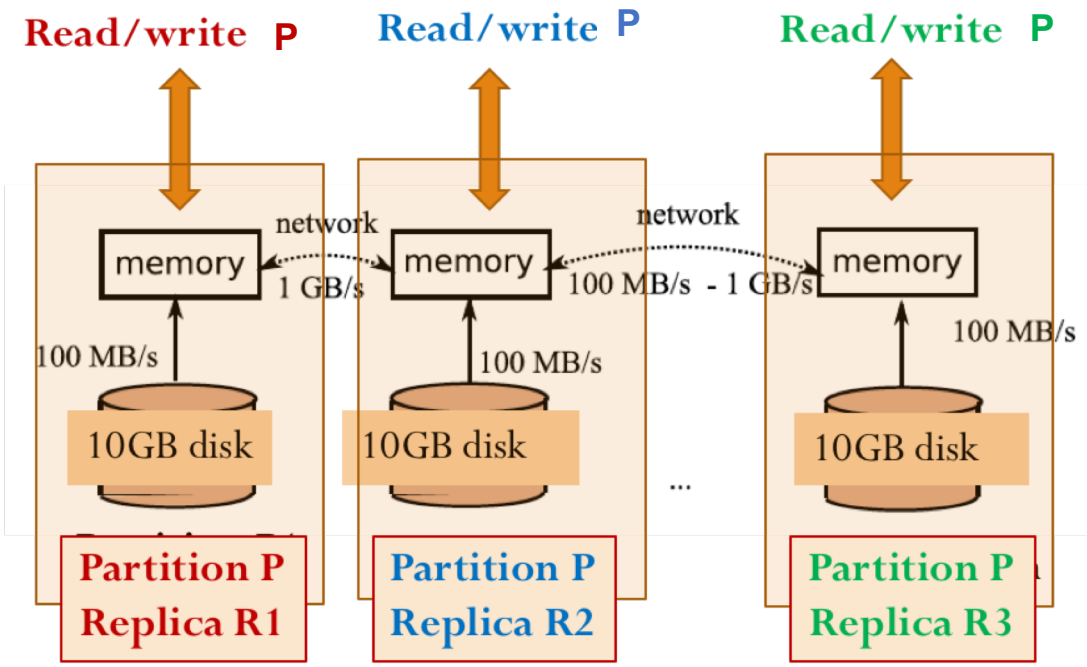


An alternative scenario – transactional application

- Workload consisting of lots of reads and writes operations, each one randomly accessing *a small piece of data in a large collection*
- More in the spirit of a database operations
- Random access to large files, seek time cannot be ignored
- Shared-nothing architectures still useful
- Distribute the load (read/write requests) through replication
  - Keeping a copy of the same data on several different nodes, potentially in different locations
  - Replication provides redundancy: if some nodes are unavailable, the data can still be served from the remaining nodes

Why load distribution?

- Shared nothing architecture
- Distributed data, distributed load
- Data replication
- Different operations executed over the same partition by different nodes



Principle #1

- Disk transfer rate is a bottleneck for batch processing of large scale data sets
  - typical of analytical processing
  - *data partitioning and task parallelization on many machines are a mean to eliminate this bottleneck*

Possible with shared nothing architectures

Principle #2

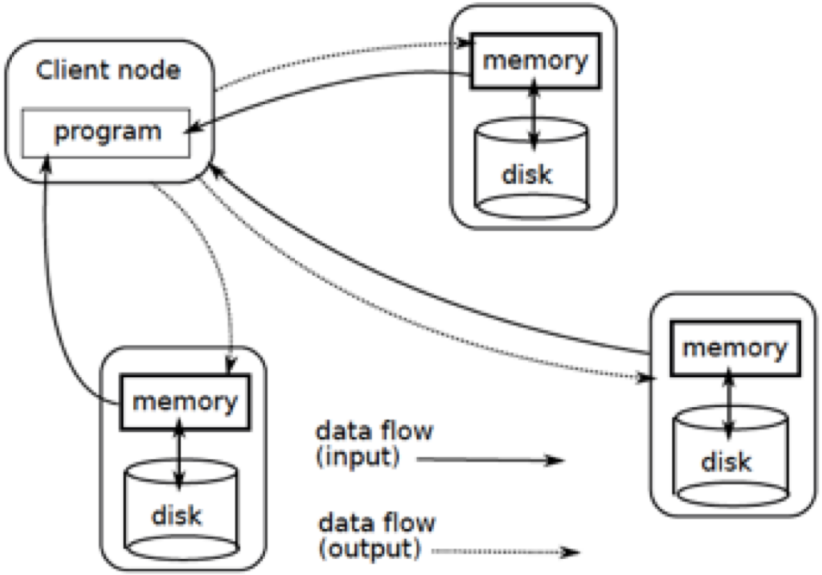
- Disk seek time is a bottleneck for transactional applications that submit a high rate of random accesses
  - typical of transaction processing
  - *replication, distribution of writes and distribution/parallelization of reads are the technical means to make such applications scalable*

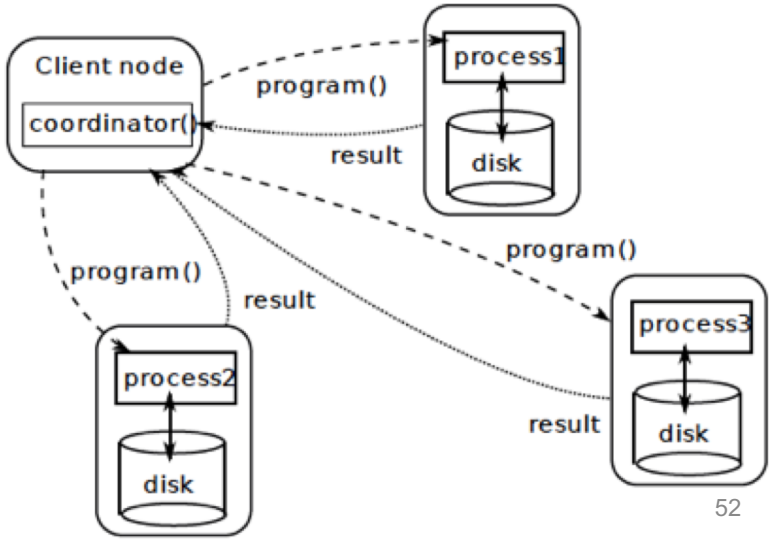
Possible with shared nothing architectures

Principle #3

- Data locality
  - bandwidth is a scarce resource, and program should be “pushed” near the data they must access to
  - in this way, we rely on more fast communications (lower costs)

Principle #3


Client node
program
memory
disk
memory
disk
memory
disk
data flow
(input)
data flow
(output)


Client node
coordinator()
process1
disk
program()
result
program()
result
process2
disk
program()
result
process3
disk

Additional motivations for data distribution – Fault tolerance/Reliability

- Reliability denotes the ability of a distributed system to deliver its services even when one or several of its software or hardware components fail
- Faults - system components deviating from their specification: program bugs, human errors, hardware or network problems
- Systems that anticipate faults and can cope with them are called fault-tolerant: at some time, they will produce the correct result
- Clusters proportionally increase the rate of hardware faults
- *Fault tolerance becomes a fundamental issue in distributed architectures*
- Fault tolerance is based on the assumption that a participating machine affected by a failure can always be replaced by another one, and not prevent the completion of a requested task
- *Reliability relies on redundancy/replication of both the software components and data*

Additional motivations for data distribution – Performance/Availability

- Availability is the capacity of a system to limit as much as possible its latency
- Involves several aspects:
- Failure detection
  - Periodically monitor the participating nodes to detect failures as early as possible
  - design quick restart protocols
- Quick recovery procedure

Said in other terms: node failures do not prevent survivors from continuing to operate

Additional motivations for data distribution - Scalability

- Scalability is the ability of a system to cope and continuously evolve in order to support increased load

- Load parameters depend on the specific system/application:
  - data volume
  - number of works (e.g., number of transactions)
  - …
- Some systems are elastic
  - They can automatically add computing resources when they detect a load increase
  - Useful if the load is unpredictable
- Some other systems are manually scaled
  - A human analyzes the capacity and decides to add more machines to the system