http://www.disit.org/6943

Big Data overview e Architetture Parallele

Prof. Paolo Nesi
DISIT Lab
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze
Via S. Marta 3, 50139, Firenze, Italia
tel: +39-055-2758515,
fax: +39-055-2758570
http://www.disit.dinfo.unifi.it
paolo.nesi@unifi.it

BY NC ND
SOME RIGHTS RESERVED

Agenda

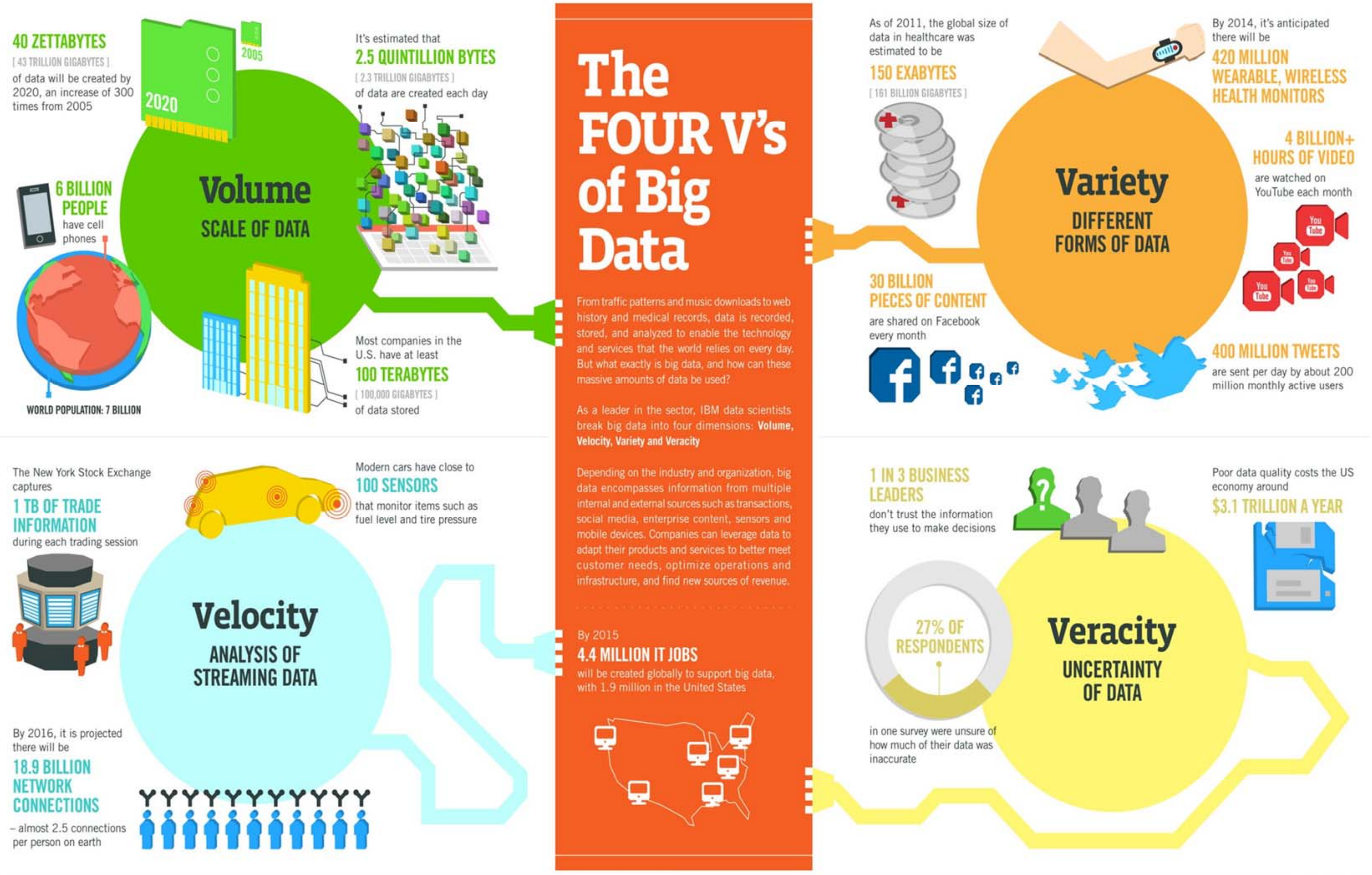
- Big Data problems
- Big Data Pipeline of working
- Distributed and Parallel Architecture
- The grid for distributed computing
- Map Reduce Approach
- CAP Theorem
- Considerations

Aree di Intervento

UNIVERSITÀ DEGLI STUDI FIRENZE
MABIDA
MASTER BIG DATA ANALYTICS AND TECHNOLOGIES FOR MANAGEMENT
GOVERNARE LA DECISIONE NELL'ONDA DEI DATI

DISIT lab slides
http://www.disit.org/6943

- Web crawling, XML, analisi dei testo
- Twitter Vigilance: NLP, sentiment analysis
- Big data introduzione
- Architetture parallele
- Reasoning and inferential
- Social Media: user profiling, recommendations
- Human behaviour
- NoSQL, graph database, ..
- Casi di studio: smart city
- Casi di studio: Smart Cloud



The FOUR V's of Big Data

From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?

As a leader in the sector, IBM data scientists break big data into four dimensions: **Volume, Velocity, Variety and Veracity**

Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States

Volume

SCALE OF DATA

40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005

It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day

6 BILLION PEOPLE
have cell phones

WORLD POPULATION: 7 BILLION

Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored

Variety

DIFFERENT FORMS OF DATA

As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]

By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS

4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month

30 BILLION PIECES OF CONTENT
are shared on Facebook every month

400 MILLION TWEETS
are sent per day by about 200 million monthly active users

Velocity

ANALYSIS OF STREAMING DATA

The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session

Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure

By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth

Veracity

UNCERTAINTY OF DATA

1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions

Poor data quality costs the US economy around
$3.1 TRILLION A YEAR

27% OF RESPONDENTS
in one survey were unsure of how much of their data was inaccurate

Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS

IBM

5V of Big Data…or more?

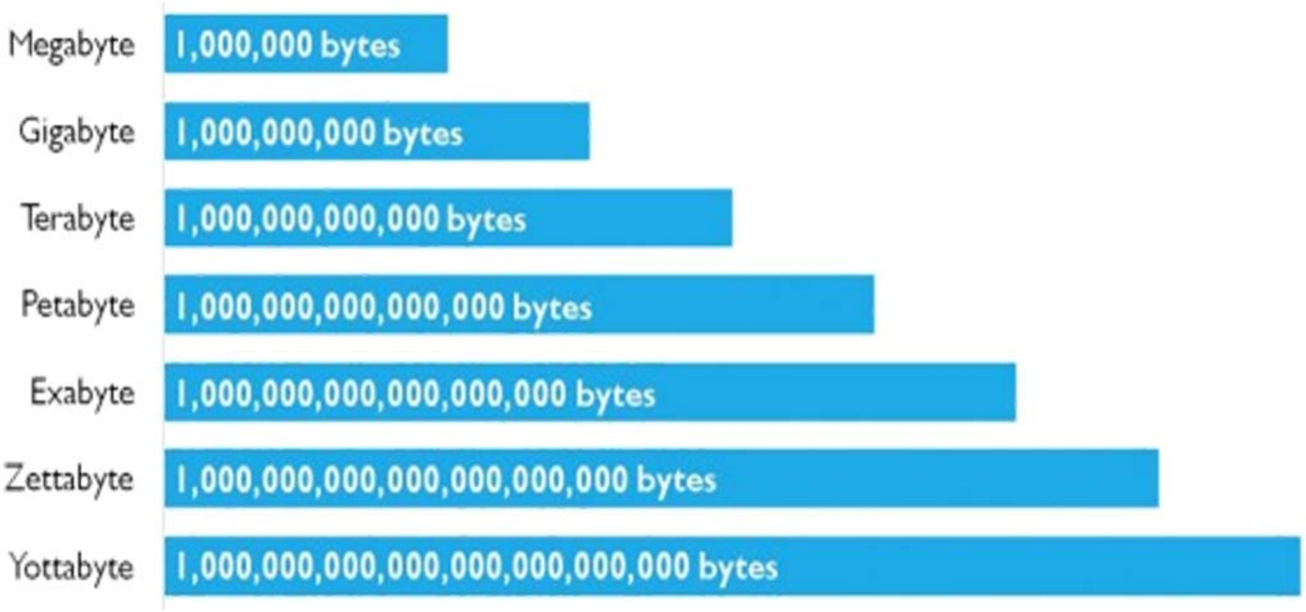
In **2010** it was estimated a production of **1.2 zettabytes** of data (**1ZB = one trillion GB**).

In **2011**, grew up in **1,8ZB**.

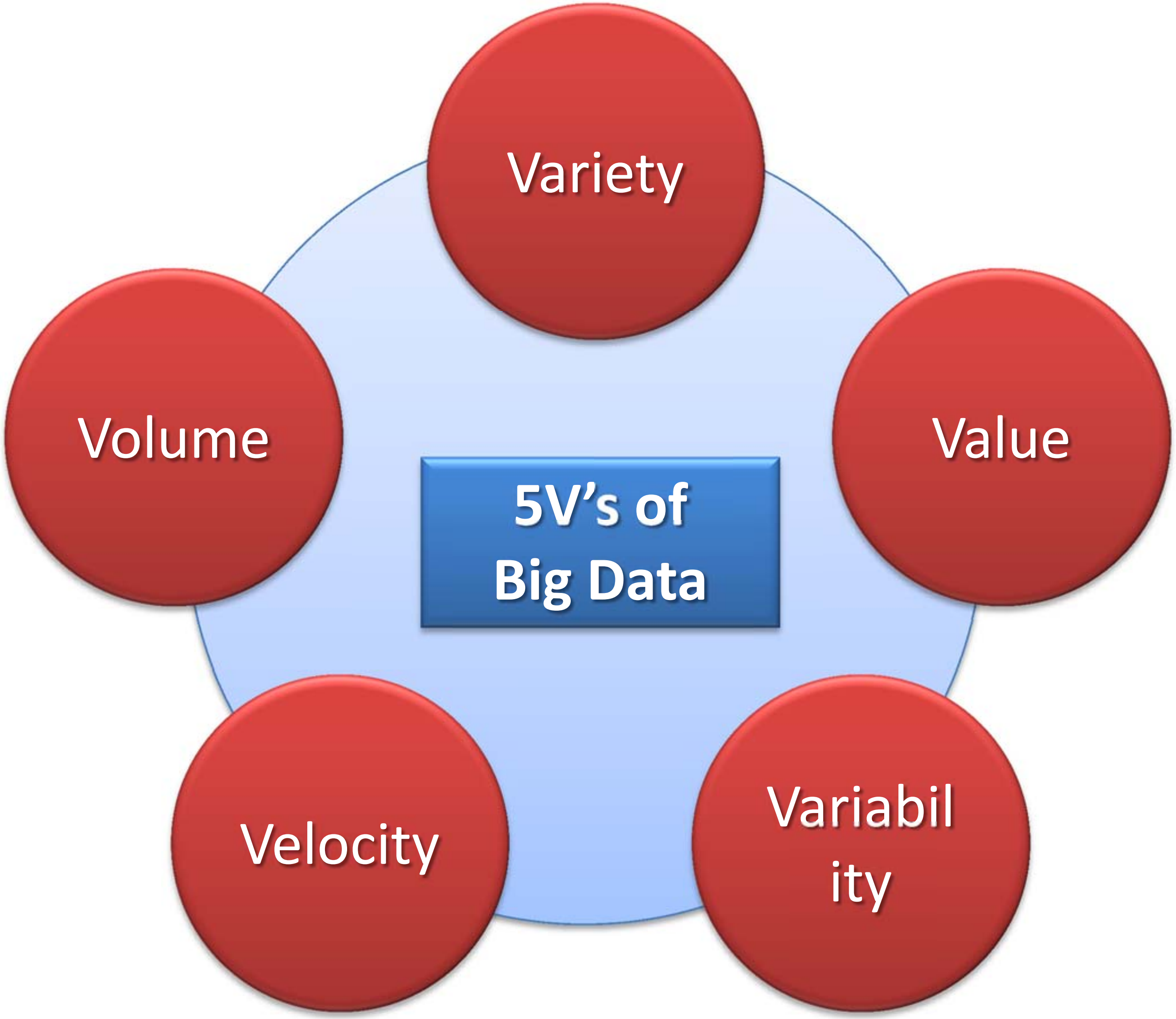
In **2013** came to **2,7ZB**.

The prediction for **2015** is about **4,8ZB**.

- Bytes and not bit
- Bytes and not structures



5V of Big Data



5V of Big Data - Velocity

- 2 meanings:
  - It refers to the **high frequency** at which **data** are **generated** and affects the amount (**volume**).
  - It refers to the **speed** at which new technologies allow **to access and analyze** this **data**.
- For **time-sensitive processes**, Big Data must be capable to perform computation in real time to maximize the **value**.

➡ Higher speed in data access

➡ Higher speed in computing, decision-making

➡ Higher market competitiveness

5V of Big Data - Velocity

- A parallel and distributed architecture is recommended:
  - Management of complex data structures.
  - Access to real-time data.
  - Good processing speed through techniques of parallel and distributed computing
  - Non-relational databases such as DB column and key/value database (NoSQL).

5V of Big Data - Variety

- **the form** in which **data** are provided.
- Big Data includes any type of data: **structured** and **unstructured data** such as **text, sensor data**, **audio, video, click streams, log files, profiles, ….**
  - Different: formats, sources, IDs, coding, etc..
- **Not suitable** to be processed with **traditional techniques** of **relational databases**: email, images, video, audio, text strings that give meaning **can not be stored in a table**.
- A **NoSQL database** is recommended: do not impose a rigid scheme to organize data (**schemaless** database)

5V of Big Data - Variability

- 2 meanings:
  - Refers to **variance** in **meaning** and in **lexicon**, that is the **data contextualization**.
  - Refers to the **variability** in **data structure**.
  - Sparse data, incomplete, etc..,
  - missing info → context dependent,
- Example: "**read the book**“
  - **positive meaning** in a blog about literature
  - **negative connotation** in a blog for movie fans.
- It is important to **find mechanisms** that are able to **give a semantics** to the data **based on the context** in which they are expressed.

5V of Big Data - Value

- Big Data **hiding** a great **value**.
- Exploiting the primary use you can extract only a part, the remaining **value remains "dormant"** until their secondary use.
- **Value is** all that you can gain from all possible modes of use of the data, the sum of many small parts that are **slowly discovered**.
- **Changing direction**: once **data were eliminated** after the first use.

It is important to adopt methods and technologies that allow **a continuous integration of new information**, following a repeated use, with the goal of **building a knowledge base** always wider

5V of Big Data…or more?

- Initially there were 3V, then 4 and now 5V of Big Data.
  - Different meanings are attributed to the 5V.
- **Viral**: refers to how much and how the data are spread (data propagation).
- The **large amount** of data and the **high speed** at which they are produced involves a **viral spread** of information.
- **Viral** is the **volume growth** of data generated by users digital activities (**user-generated content**)

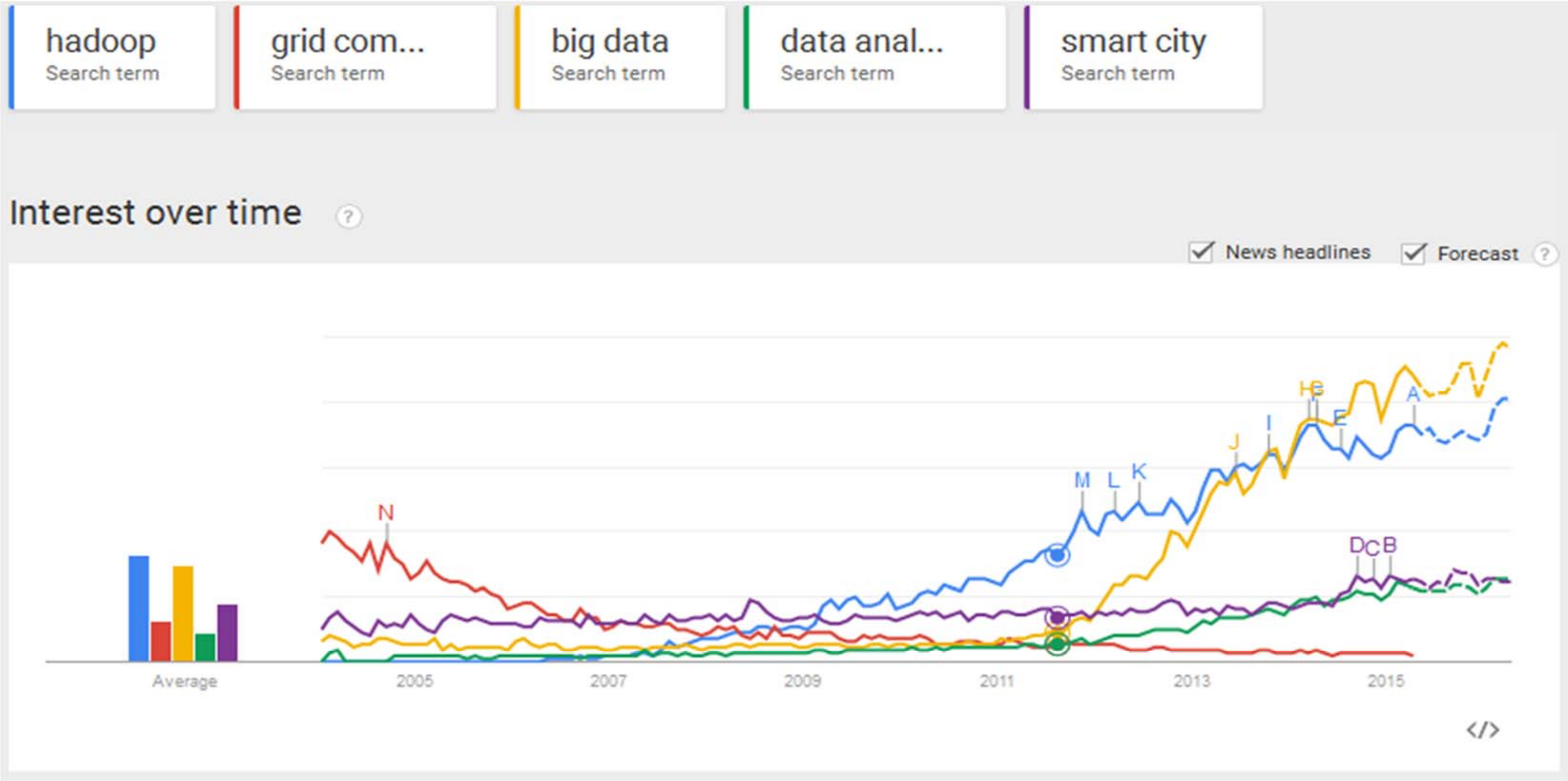
Why a content became viral?

- A data analysis company sought to understand what are the characteristics that make a viral content:
  - **Dimensions**: greater length of content → more shares.
  - **Emotions**: people love to share elements that cause laughter and amazement (42%). Emotions to avoid: sadness and fear (7%).
  - **Images**: visual contents attract attention, encourage understanding → increase shares on social. 65% of people use Facebook to share posts with at least 1 image. More than 20% of Twitter users prefer to publish content with an image.

Why a content became viral?

- A data analysis company sought to understand what are the characteristics that make a viral content:
  - **Bulleted lists**: web users love bulleted lists, infographics, and how-to, because they allow to summarize salient aspects in visual form, making them easy to understand.
  - **Influencer**: content shared by people considered "experts" reach a greater number of users, "targeted" and interested.

Google Trends



hadoop
Search term

grid com...
Search term

big data
Search term

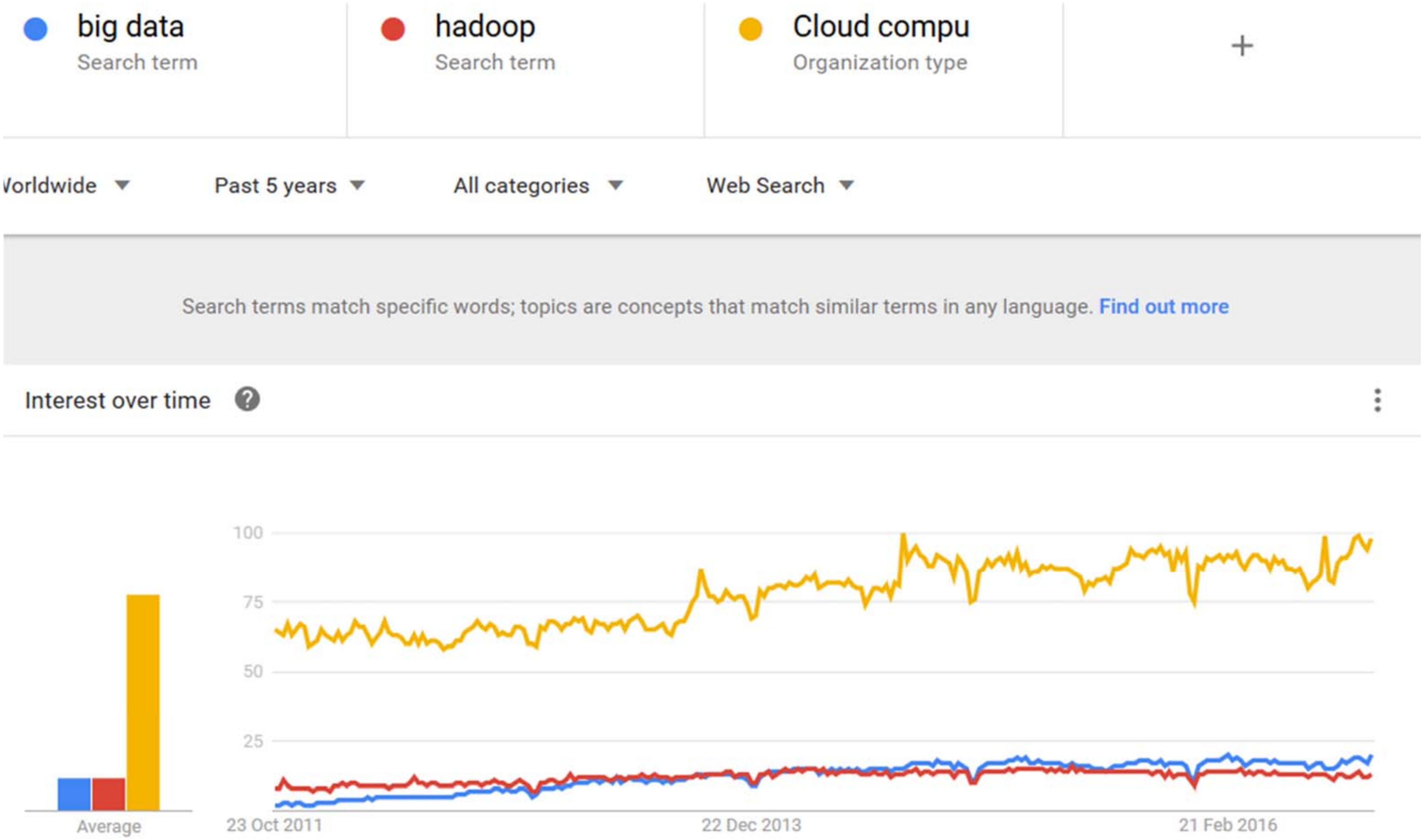
data anal...
Search term

smart city
Search term

Interest over time ?

☑ News headlines ☑ Forecast ?

Google Trends



● big data
Search term

● hadoop
Search term

● Cloud compu
Organization type

+

Vorldwide ▼ Past 5 years ▼ All categories ▼ Web Search ▼

Search terms match specific words; topics are concepts that match similar terms in any language. Find out more

Interest over time

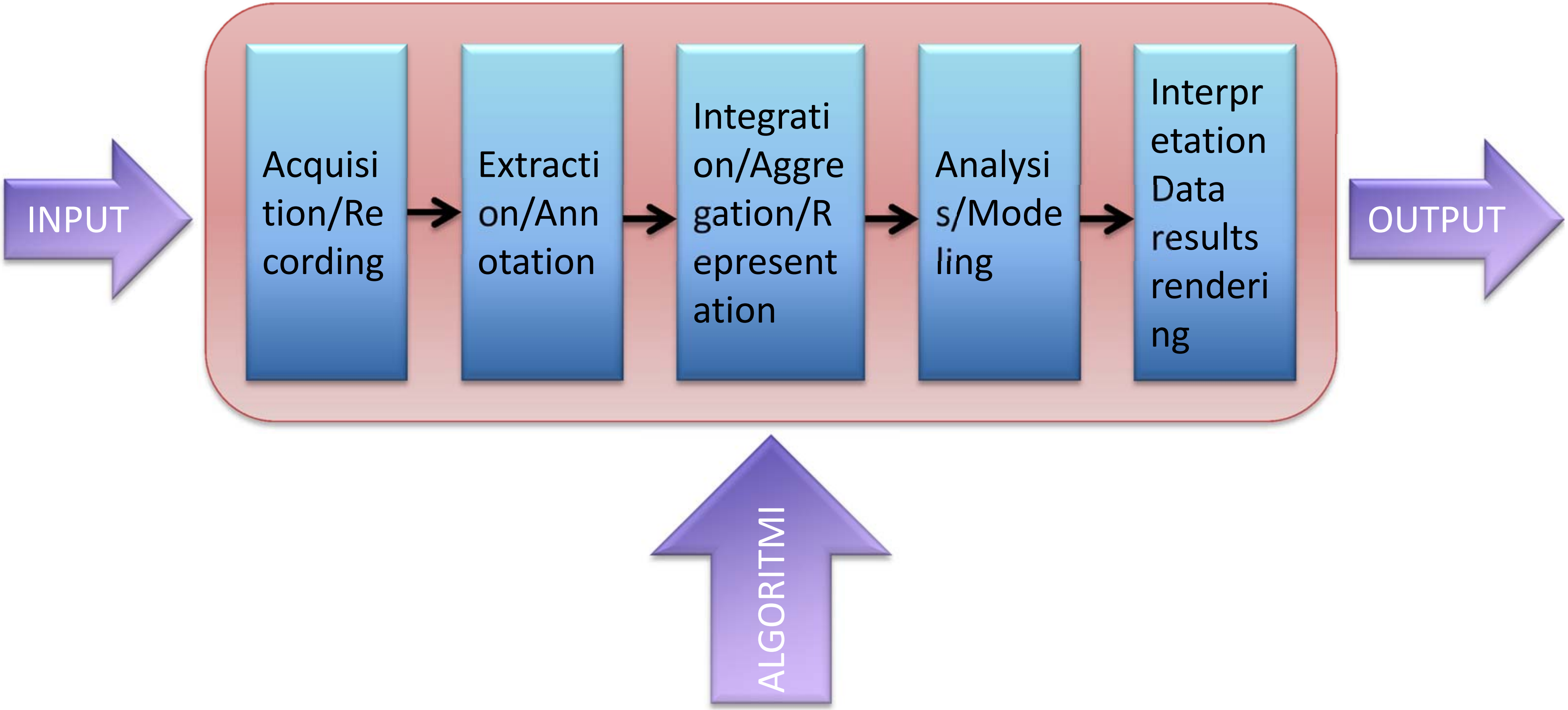
Questi Big data problem ?

- Siamo nell’ambito dei Big Data quando le dimensioni sono parte del problema
  - Relativo vs assoluto ?
- Come accorgersene: i metodi tradizionali / commerciali e allo stato dell’arte non funzionano
  - Non danno risposte in tempi ragionevoli
  - Non danno risposte perché si arriva ai loro limiti….
    - Memoria, velocita’, storage, etc.
- **Sono necessarie soluzioni che avendo solo dati small non sarebbero utilizzate,**
  - **perché magari sono approssimate, troppo costose, complesse da usare, solo per professionisti, ….**

Agenda

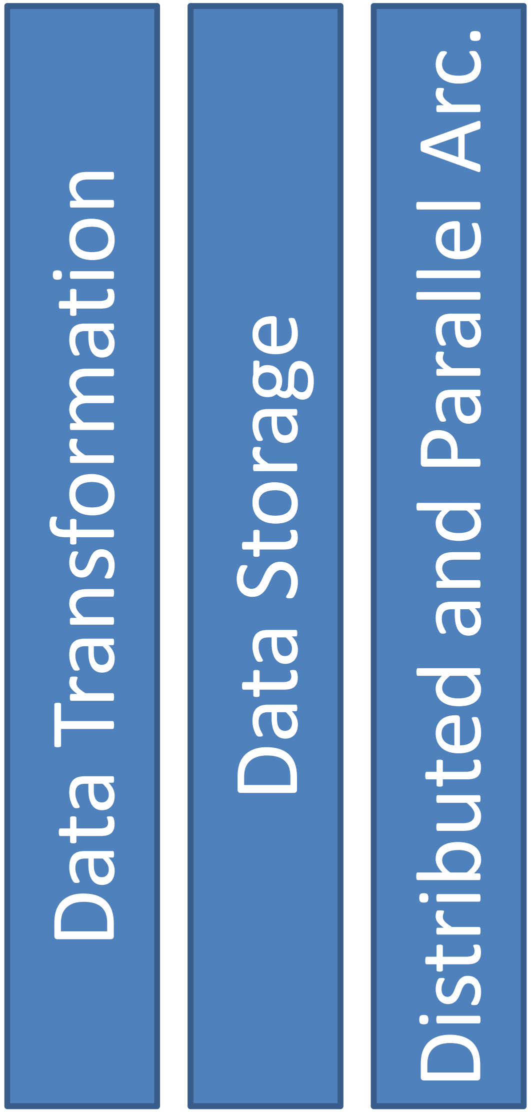
- Big Data problems
- Big Data Pipeline of working ←
- Distributed and Parallel Architecture
- The grid for distributed computing
- Map Reduce Approach
- CAP Theorem
- Considerations

Pipeline


INPUT
Acquisition/Recording
Extraction/Annotation
Integration/Aggregation/Representation
Analysis/Modeling
Interpretation Data results rendering
OUTPUT
ALGORITMI

Data Transformation

- **Data gathering**
  - Harvesting, crawling, streaming, ….
- **Data extraction,** early mining
  - filtering, completing, quality improvement, …
  - Natural language processing
  - noise vs signal/information
- **Data aggregation**
  - reconciliation
  - semantic vs tabular



Information Extraction & Cleaning

- The **information collected** will **not** be **in** a format ready for **analysis** (surveillance photo VS picture of the stars).
  - **Challenge**: realize an information extraction process that pulls out information, and express it in a form suitable for analysis.
- **Big Data** are **incomplete** and **errors** may have been committed during **Data Acquisition** phase.
  - **Challenge**: define constraints and error models for many emerging Big Data domains.
  - ***Be careful of cutting out the searched information***

Data Integration, Aggregation, Representation

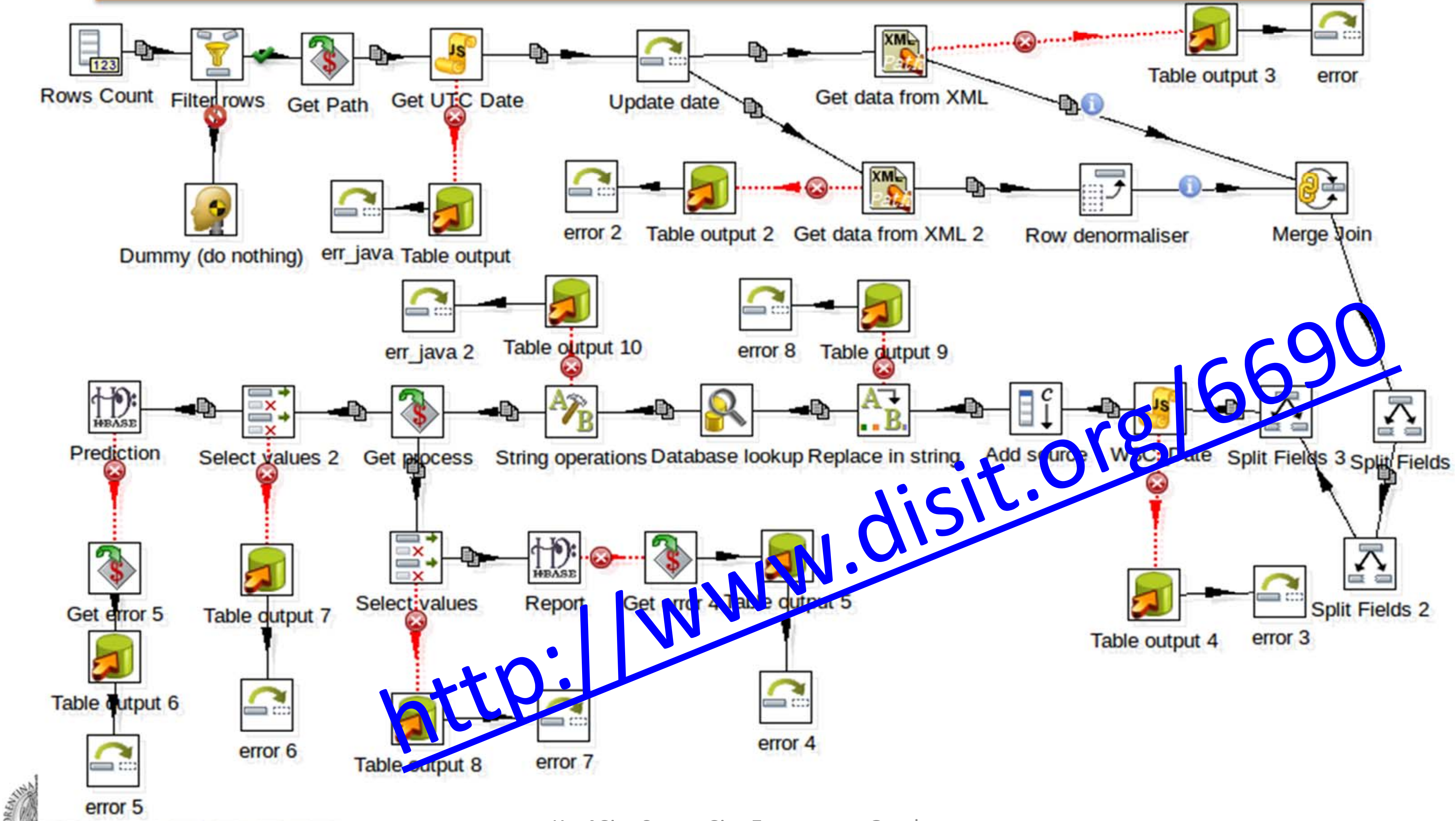
- Data is **heterogeneous**, it is not enough throw it into a repository.
  - **Challenge**: create a data record structure that is suitable to the differences in experimental details.

- **Many ways to store** the same information: some designs have advantages over others, for certain purposes.
  - **Challenge:** create tools to assist in database design process and developing techniques.

Query Processing, Data Modeling & Analysis

- Methods for querying and **mining Big Data** are **different** from traditional **statistical analysis**.
  - **Challenge**: create a scaling complex query processing techniques to terabytes while enabling interactive response times.
- **Interconnected Big Data** forms large **heterogeneous networks**, with which **information redundancy** can be explored to **compensate** for **missing data**, to **crosscheck conflicting** cases and to uncover hidden relationships and models.
  - **Challenge**: add coordination between database systems and provide SQL querying, with analytics packages that perform various forms of non-SQL processing (data mining, statistical analysis).

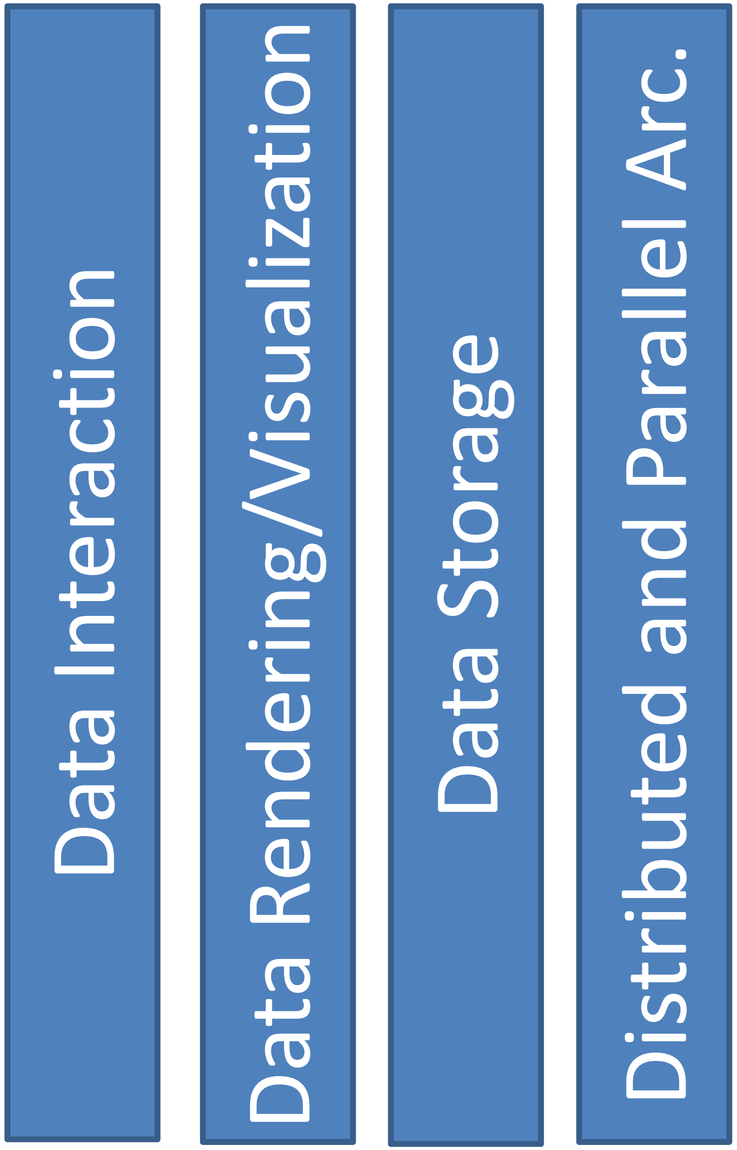
Example of Ingestion process



http://www.disit.org/6690

Hot issues in the pipeline

- **Data Storage vs Modeling**
  - Needed or not needed ?
  - Data model: tabular vs graph
  - CAP theorem, HA, Recovering, …
- **Data Analytics/Mining**
  - Data analysis
    - Finding, discovering, black swans,
    - unexpected results, Incidental finding
  - Modeling vs business value:
    - Prediction, early warning, …..
- **Visual Analytics**
  - Real time computing vs precomputing



Data Interaction

Data Rendering/Visualization

Data Storage

Distributed and Parallel Arc.

Datification

- **Datification**: to taking information about all things and transforming it into a data format to make it quantified.
  - *See for example Twitter Vig: from TWs to metrics*
- Use this information in new ways to **unlock** the implicit, latent **value** of this information.
- When the data were "**few**", it was desirable they were **accurate** (Random Sampling).
  - BigData have changed the expectations of precision: to deal with these **large quantities** of data as something imprecise and imperfect allows us to make superior forecasts (**Predictive Analysis**).
  - 5 V are part of the problem: noise, variability, incompleteness and inconsistencies.

Big Data Solutions

- Considering technologies and solutions for Big Data, **4 correlated aspects** have to be addressed:
  - Data Management aspects;
  - Architectural aspects;
  - Access/Data Rendering aspects;
  - Data Analysis and Mining/Ingestion aspects.

Data Management aspects

- **Scalability**: Each storage system for big data must be scalable, with common and cheap hardware (increase the number of storage discs.)
- **Tiered Storage**: To optimize the time in which we want the required data.
- **High availability**: Key requirement in a Big Data architecture. The design must be distributed and optionally lean on a cloud solution.
- **Support to Analytical and Content Applications**: analysis can take days and involve several machines working in parallel. These data may be required in part to other applications.
- **Workflow automation**: full support to creation, organization and transfer of workflows

Data Management aspects

- **Integration with existing Public and Private Cloud systems**: Critical issue: transfer the entire data. Support to existing cloud environments, would facilitate a possible data migration.
- **Self Healing, resilience**: The architecture must be able to accommodate component failures and heal itself without customer intervention. Techniques that automatically redirected to other resources, the work that was carried out by failed machine, which will be automatically taken offline.
- **Security**: The most big data installations are built upon a web services model, with few facilities for countering web threats, while it is essential that data are protected from theft and unauthorized access.

Architectural aspects

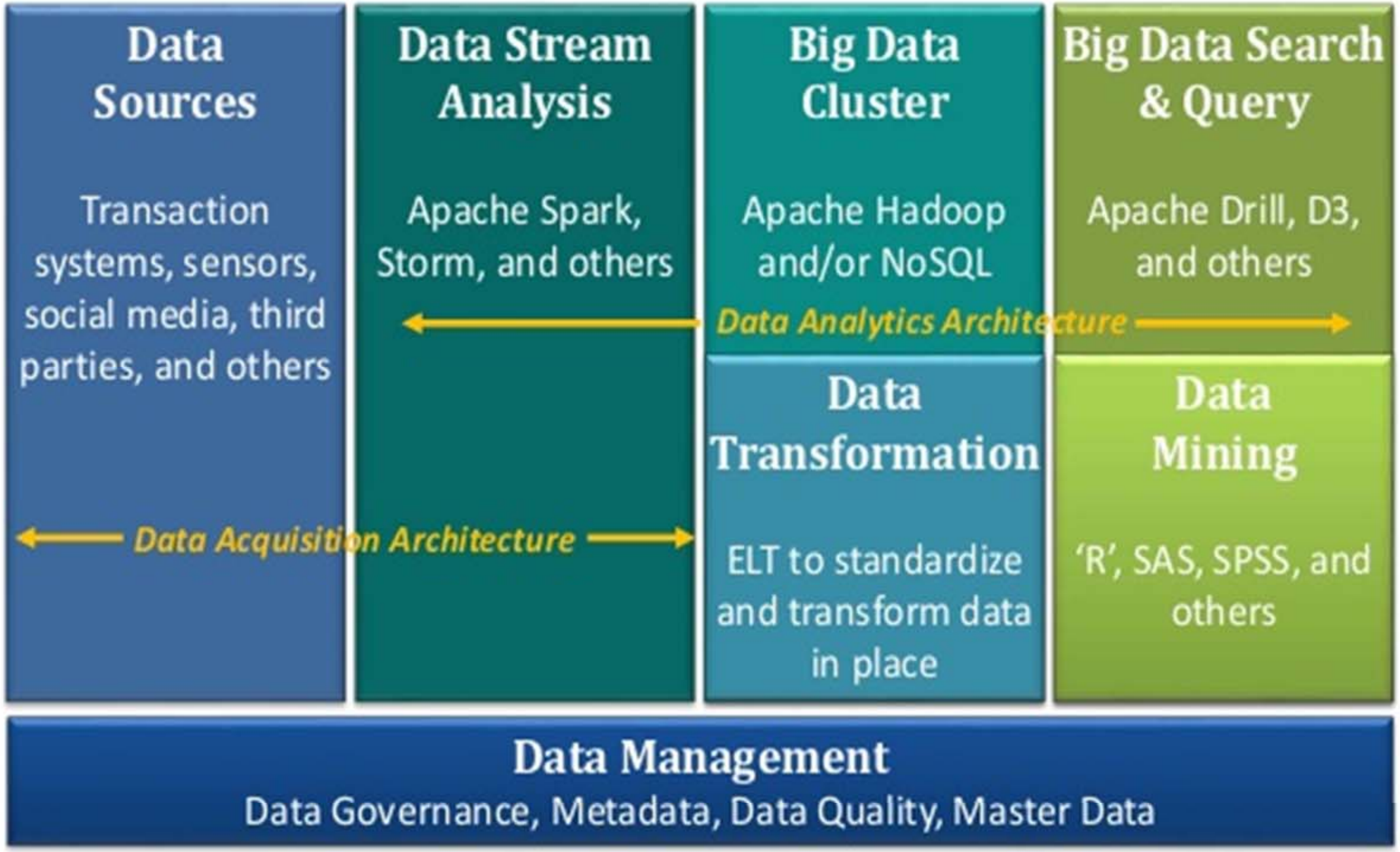
- **Clustering size**: the number of nodes in each cluster, affects the completion times of each job: a greater number of nodes, correspond to a less completion time of each job.
- **Input data set**: increasing the size of the initial data set, the processing time of the data and the production of results also increase.
- **Data node**: greater computational power and more memory are typically associated with a shorter time to completion of a job → **dualism memory and velocity**
- **Date locality**: it is not possible to ensure that the data is available locally on the node. we need to retrieve the data blocks to be processed, and the time of completion be significantly higher. **Velocity of the massive storage**

Architectural aspects

- **Network**: The network affects the final performance of a Big Data management system;
  - connections between clusters make extensive use during read and write operations.
  - Highly available and resiliency network, which is able to provide redundancy.
- **Cpu**: more processes to be carried out are CPU-intensive, greater will be the influence of the CPU power on the final performance of the system.
  - Many tool are not capable to exploit!
- **Memory**: For applications memory-intensive is good to have the amount of memory on each server, able to cover the needs of the cluster. 2-4GB of memory for each server, if memory is insufficient performance would suffer,. But may depend on the general architecture and approach .

ig Data High-level Architecture

RCG GLOBAL SERVICES®



Data Sources

Transaction systems, sensors, social media, third parties, and others

Data Stream Analysis

Apache Spark, Storm, and others

Big Data Cluster

Apache Hadoop and/or NoSQL

Big Data Search & Query

Apache Drill, D3, and others

Data Analytics Architecture

Data Transformation

ELT to standardize and transform data in place

Data Mining

'R', SAS, SPSS, and others

Data Acquisition Architecture

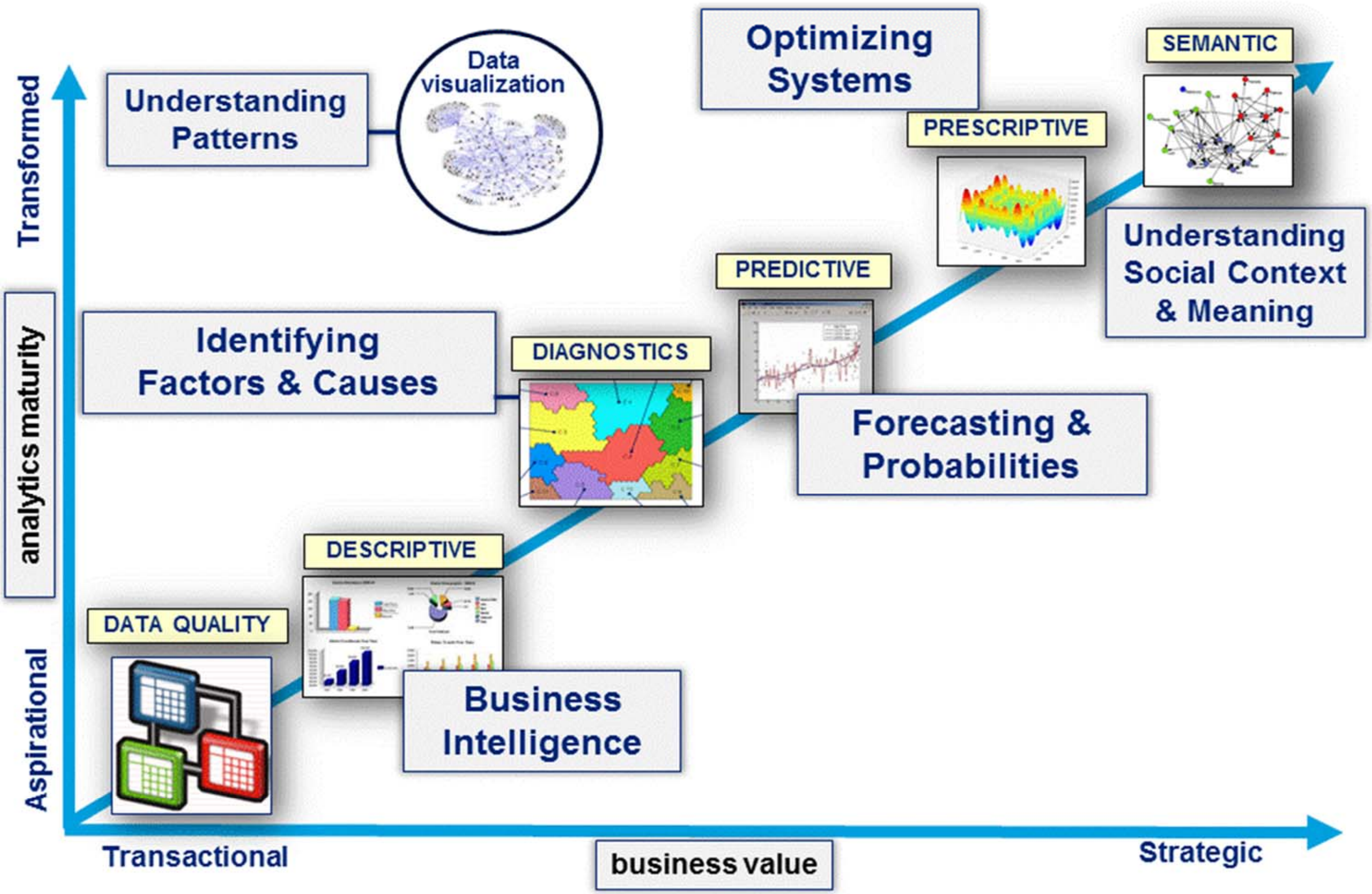
Data Management

Data Governance, Metadata, Data Quality, Master Data

+

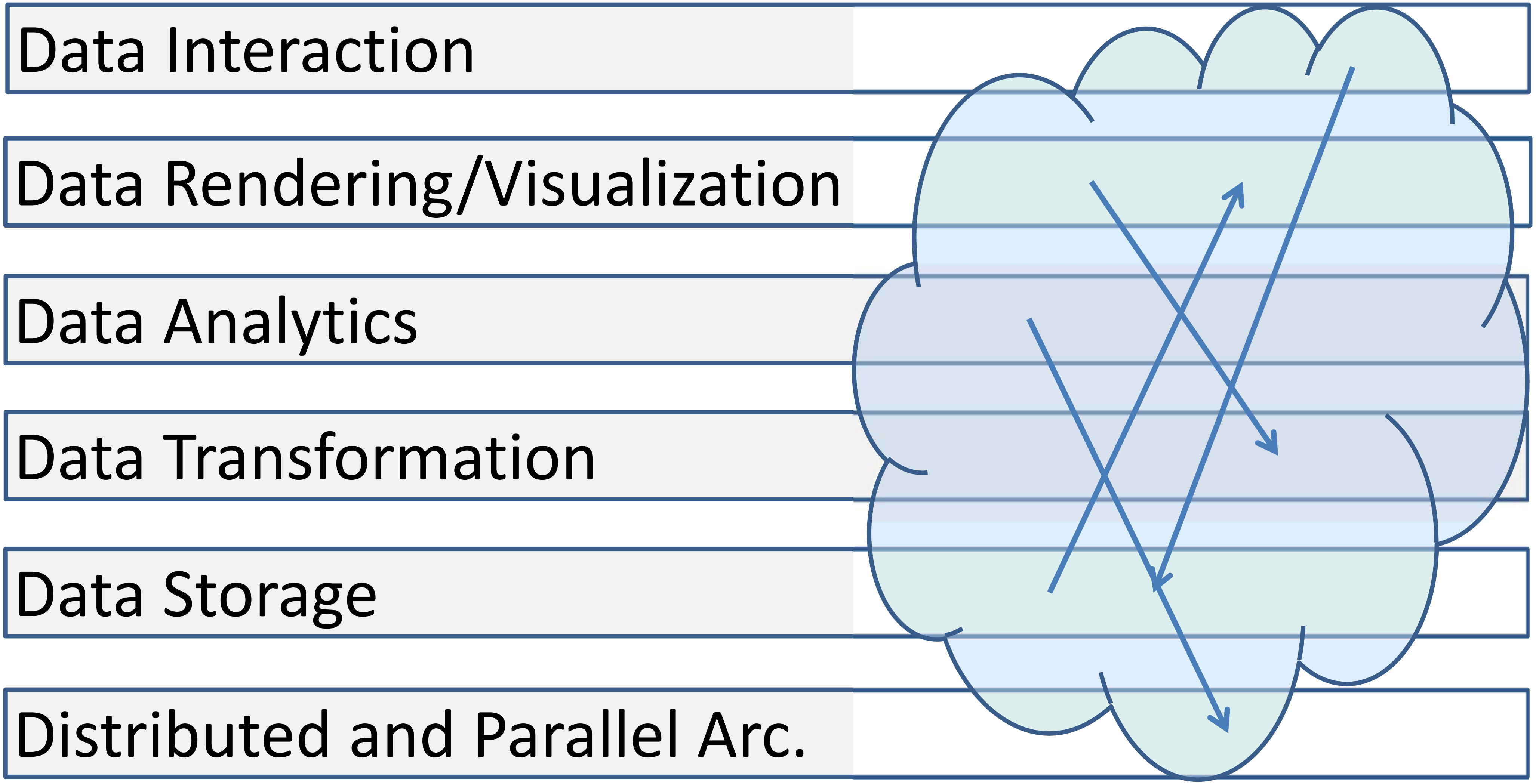
Data Rendering and tooling:

Data Rendering in Real Time


Understanding Patterns
Data visualization
Optimizing Systems
SEMANTIC
PRESCRIPTIVE
Understanding Social Context & Meaning
PREDICTIVE
Identifying Factors & Causes
DIAGNOSTICS
Forecasting & Probabilities
DESCRIPTIVE
DATA QUALITY
Business Intelligence
Transformed
analytics maturity
Aspirational
Transactional
business value
Strategic

Lo stack

dipendenze



Data Interaction

Data Rendering/Visualization

Data Analytics

Data Transformation

Data Storage

Distributed and Parallel Arc.

Agenda

- Big Data problems
- Big Data Pipeline of working
- Distributed and Parallel Architecture ←
- The grid for distributed computing
- Map Reduce Approach
- CAP Theorem
- Considerations

Distributed and Parallel Architecture

- Siamo abituati a utilizzare sistemi a microprocessore: calcolatori elettronici … ma come funzionano?
- Nel senso comune il mito che
  - se una certa elaborazione si puo' ottenere in 5 minuti potrebbe essere possibile ottenerla in 2.5 usando il doppio delle risorse..
- Questo è discutibile e va compresa meglio
  - I database (memoria permanente per contenere informazioni complesse a lungo tempo, e ricercabili in qualche modo)
  - I sistemi di calcolo (elementi a microprocessore in grado di svolgere operazioni di calcolo complesse in tempi brevi)

Il Contesto Tecnologico

Crescita delle risorse

❖ Il numero di transistor raddoppia ogni 18 mesi (Legge di Moore)

❖ La velocità dei computer raddoppia ogni 18 mesi

❖ La densità di memoria raddoppia ogni 12 mesi

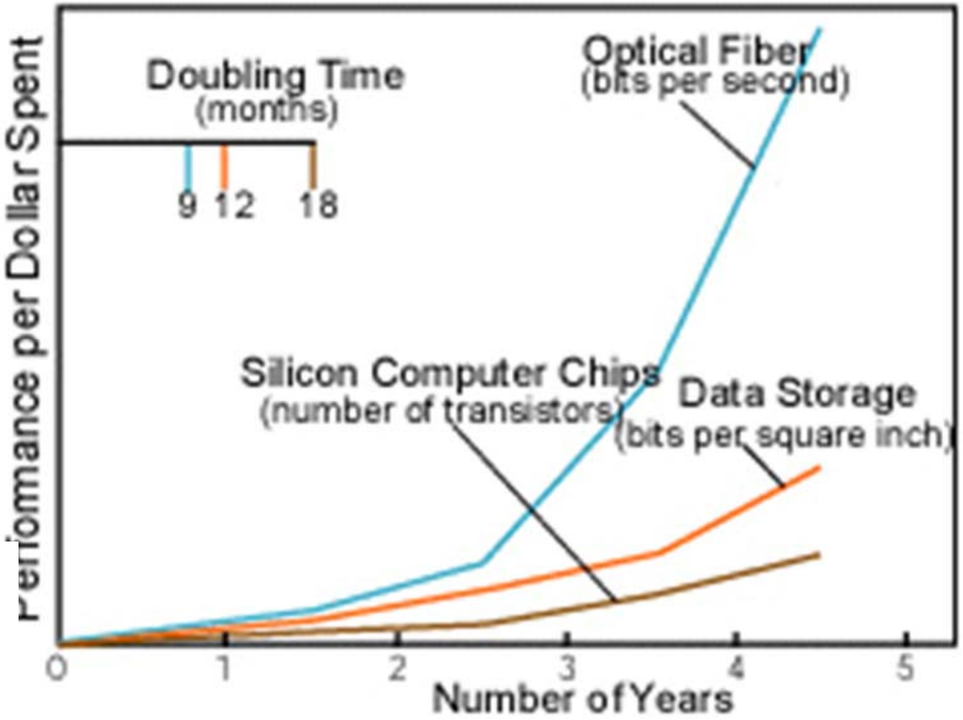
❖ La velocità della rete raddoppia ogni 9 mesi

Differenza = un ordine di grandezza ogni 5 anni

Pertanto ogni anno che passa diventa piu' conveniente usare delle soluzioni distribuite.

Network Exponentials

- Network vs. computer performance
  - Computer speed doubles every 18 months
  - Network speed doubles every 9 months
  - Difference = one order of magnitude every 5 years
- 1986 to 2000
  - Computers: x 500
  - Networks: x 340,000
- 2001 to 2010
  - Computers: x 60
  - Networks: x 4000



Moore's Law vs. storage improvements vs. optical improvements. Graph from **Scientific American** (Jan-2001) by Cleo Vilett, source Vined Khoslan, Kleiner, Caufield and Perkins.

Frieda’s Application …

Simulate the behavior of **F**(*x,y,z*) for 20 values of *x*, 10 values of *y* and 3 values of *z* (20*10*3 = 600 combinations)

- **F** takes on the average 6 hours to compute on a “typical” workstation (total = 3600 hours)
- **F** requires a “moderate” (128MB) amount of memory
- **F** performs “moderate” I/O - (*x,y,z*) is 5 MB and **F**(*x,y,z*) is 50 MB

Non posso aspettare 3600 ore

Non posso aspettare 3600 ore

- Potrei eseguire le 600 **F**(*x,y,z*) su 600 calcolatori
  - Forse va considerato anche il costo di comunicazione dei dati
- E’ Possibile allocare le 600 computaizoni di **F**(*x,y,z*) se sono esecuzioni completamente indipendenti (come in questo caso),
  - dove ogni processo parte da dati che non dipendono dai risultati degli altri processi, delle altre esecuzioni

Ordini di grandezza

- Le **CPU eseguono** operazioni a frequenze elevate per esempio 2-3 Ghz,
  - sono bilioni di colpi di clock al secondo
  - Ogni operazione matematica necessita di decine di colpi di clock
  - Ogni algoritmo necessita di centinaia di operazioni matematiche..
  - Ogni sistema a microprocessore puo’ essere piu’ o meno efficiente nello svolgere queste operazioni, ….
- **Lo storage**, gli HD/SSD sono elementi che memorizzano in modo permanente I dati con tempi dell’ordine dei millisecondi di accesso e/o scrittura.
- Si ha grossomodo un **fattore 1000** come rapporto fra la velocita’ con la quale si esegue un accesso in memoria a quello a disco
- L’accesso a disco è tipicamente il collo di bottiglia → SSD
- Se si ha un **problema di big data** difficilmente i dati stanno tutti in memoria → l’uso efficiente di strutture dati che stanno su HD/SSD è indispensabile.

Architetture Parallele

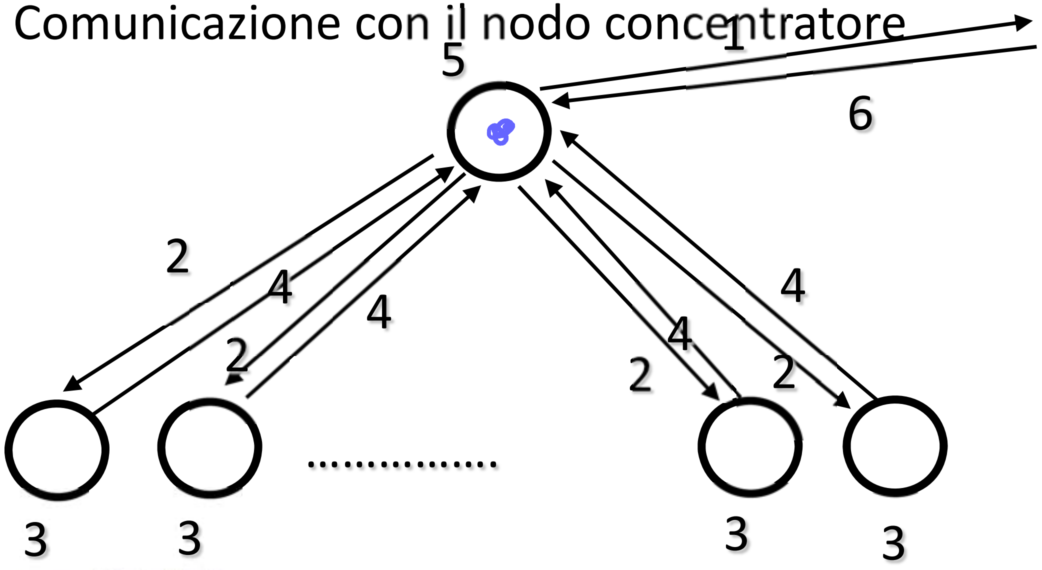
- ❑ La definizione di un’architettura ottima in termini di processi paralleli per il calcolo scientifico dipende dal problema
- ❑ Non confondiamo il Problema con la Soluzione

- ❑ Vi sono **problemi**
  - intrinsecamente sequenziali
  - Lineari vettoriali
  - Multidimensionali vettoriali
  - Paralleli nei dati di ingresso
  - Paralleli nei dai di uscita
  - Paralleli nei servizi
  - Paralleli nella procedura
  - Etc..

Esempio di caso Lineare

- VettC = VettA + VettB
- In modo sequenziale il Costo e’ o(N), in parallelo il costo e’ 1
- Soluzione parallela:
  - N nodi
  - Un concentratore per raccolta dati
  - Comunicazione fra nodi: assente
  - Comunicazione con il nodo concentratore

Per avere vettori lunghi [illegible]!

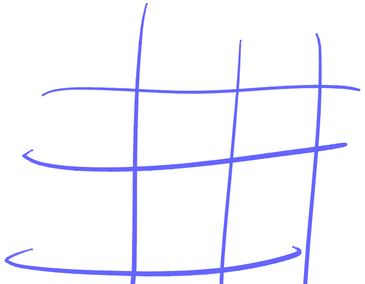


1. Passa A e B
2. Passa Ai, Bi
3. ∀i Calcola Ai+Bi
4. Passa Ci
5. Metti insieme C
6. Passa C

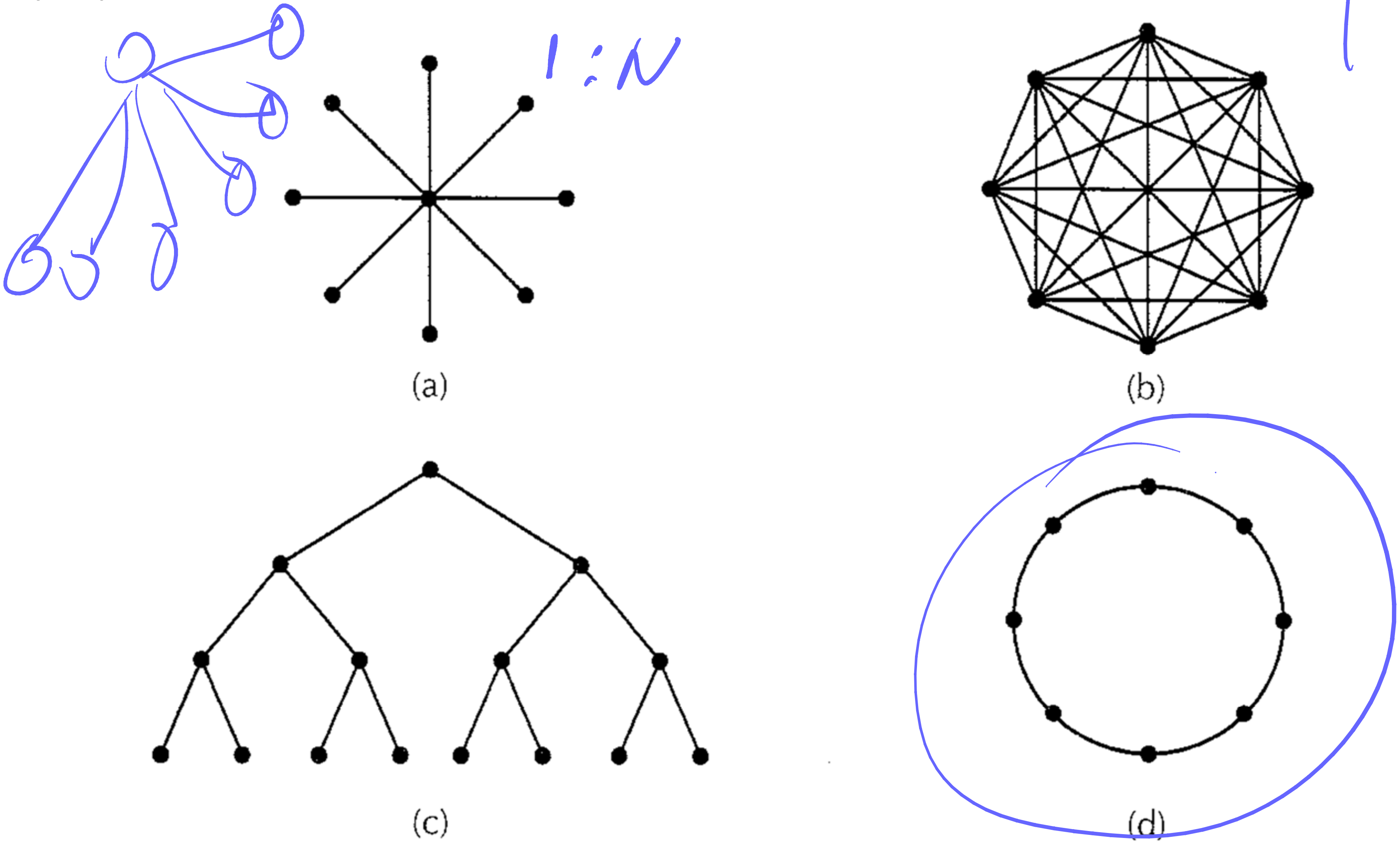
Comunicazione fra processi

- In alcuni casi vi è la necessità di effettuare connessioni/comunicazioni dirette fra modi dell'architettura parallela
- Se queste comunicazioni **possono** essere eseguite in parallelo (contemporaneamente) si risparmia tempo rispetto a farle gestire tutte da un nodo centrale come in molti sistema di gestione di processi concorrenti
- Un sistema di **gestione di processi concorrenti** dovrebbe
  - permettere di mappare in modo logico un'architettura parallela/concorrente qualsiasi architettura fisica in termini di processori e calcolatori
  - Permettere ai nodi (processori) di comunicare chiamandosi in modo logico e non fisico
  - Permettere di identificazione in modo logico i nodi.

Soluzioni parallele diverse



O(n)+O(n



(a)

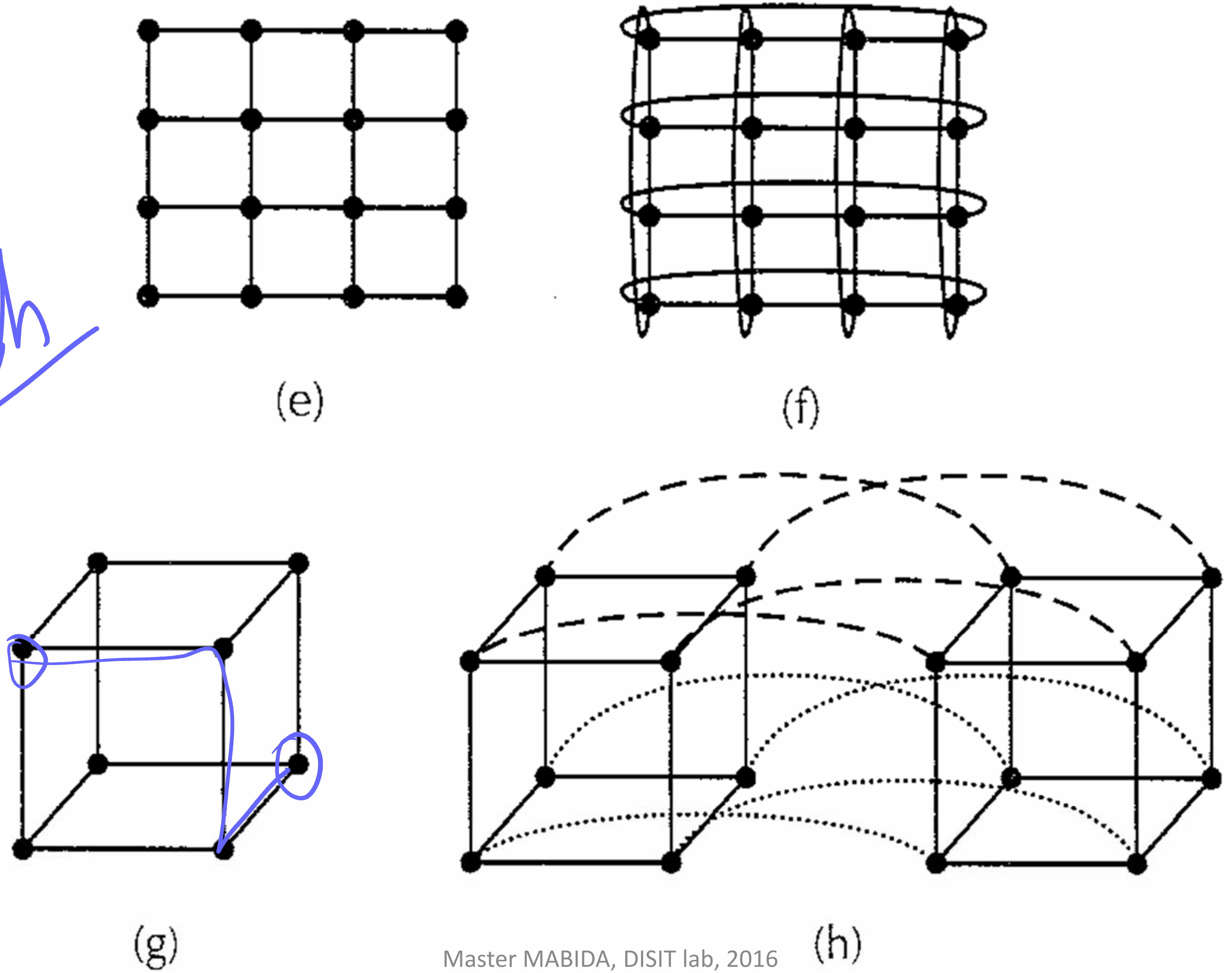
(b)

(c)

(d)

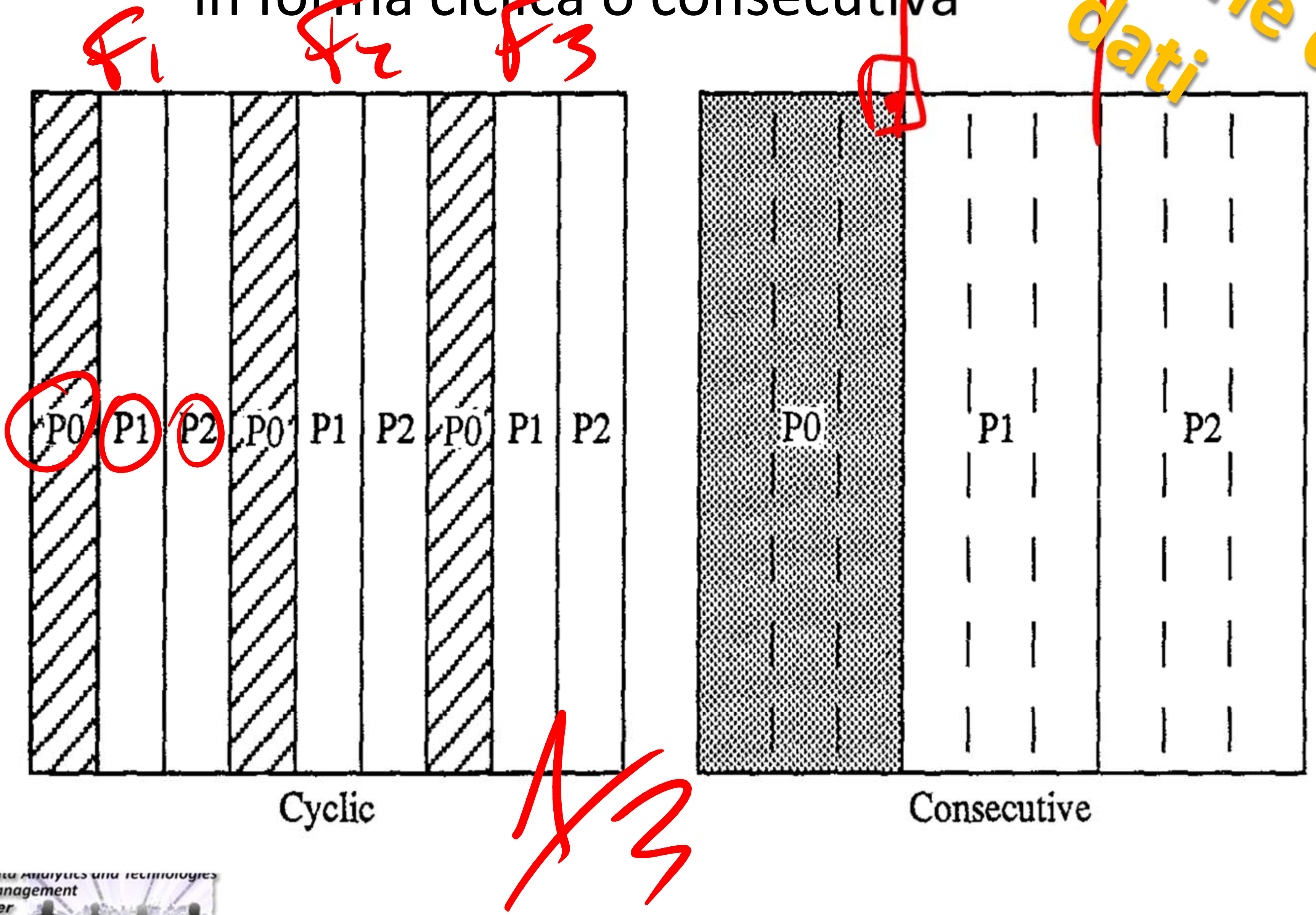
Soluzioni parallele diverse

Mesh



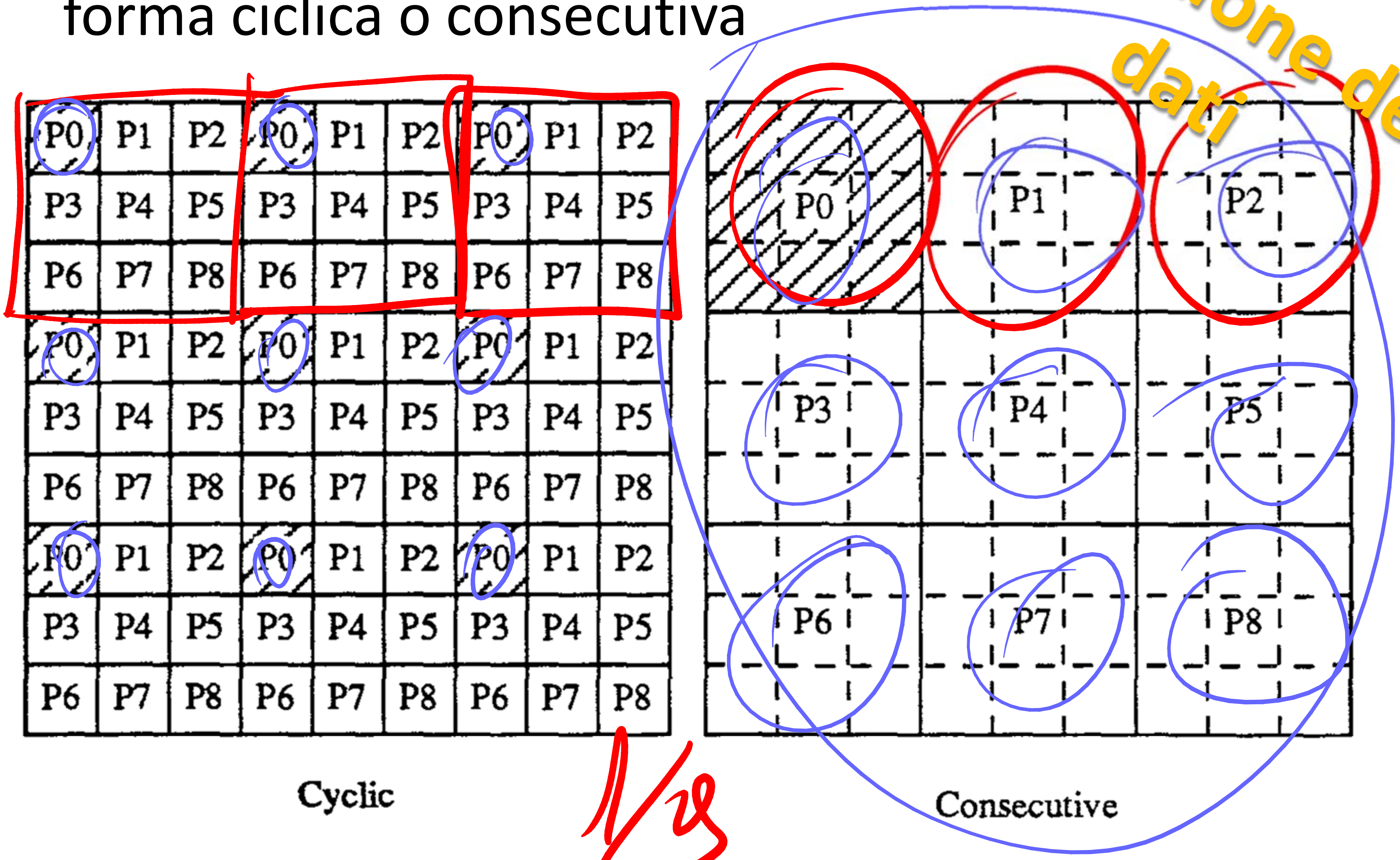
3 Processori
in forma ciclica o consecutiva

Partizione dei dati



8 Processori in forma ciclica o consecutiva

Partizione dei dati



P0	P1	P2	P0	P1	P2	P0	P1	P2
P3	P4	P5	P3	P4	P5	P3	P4	P5
P6	P7	P8	P6	P7	P8	P6	P7	P8
P0	P1	P2	P0	P1	P2	P0	P1	P2
P3	P4	P5	P3	P4	P5	P3	P4	P5
P6	P7	P8	P6	P7	P8	P6	P7	P8
P0	P1	P2	P0	P1	P2	P0	P1	P2
P3	P4	P5	P3	P4	P5	P3	P4	P5
P6	P7	P8	P6	P7	P8	P6	P7	P8

Cyclic

P0	P1	P2
P3	P4	P5
P6	P7	P8

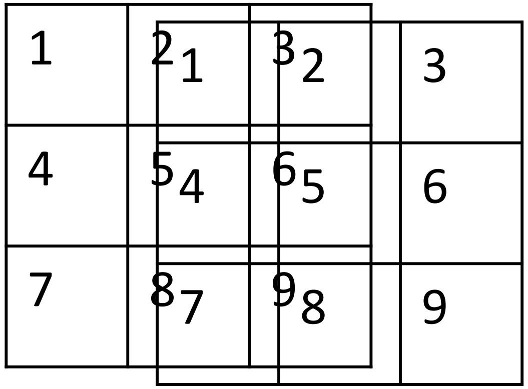
Consecutive

1/28

Esempio di elaborazione locale spaziale

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

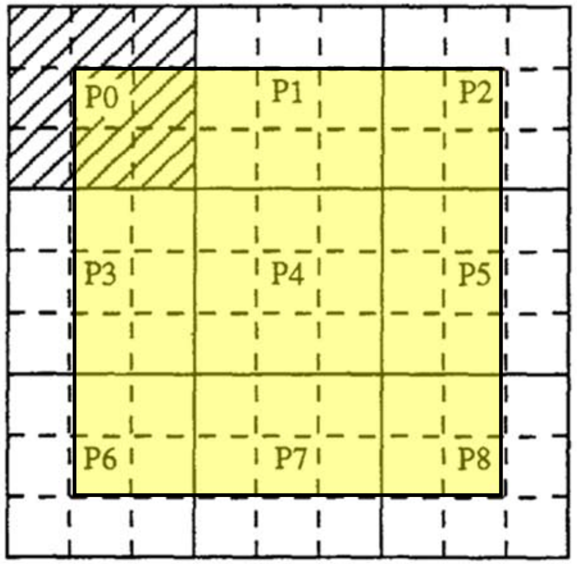


Media: $= \sum_{i=1}^{9} Matrice\ (i)$

per stimarla per ogni punto della matrice:

Problemi al contorno

Sovrapposizione dei dati

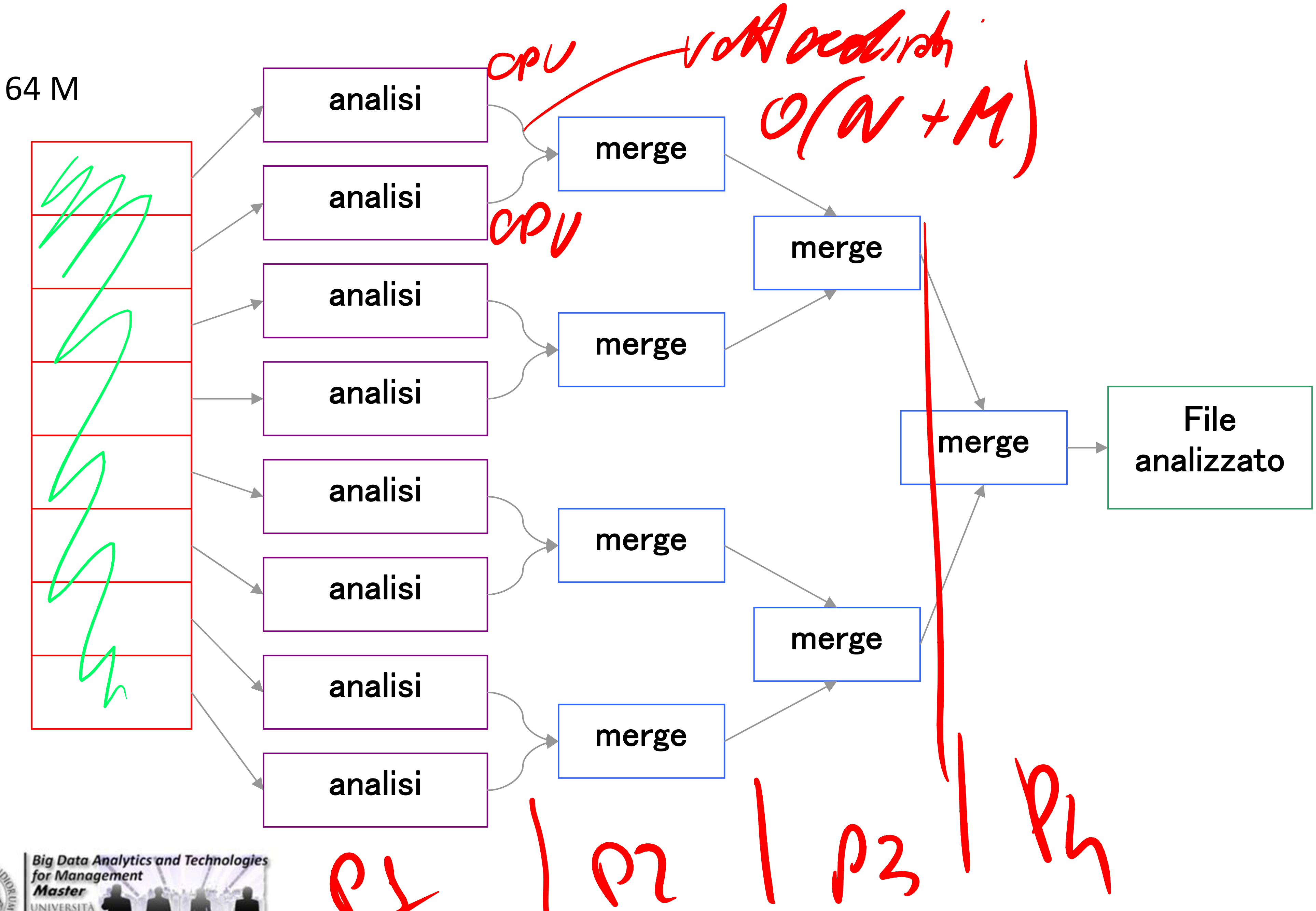


Un esempio

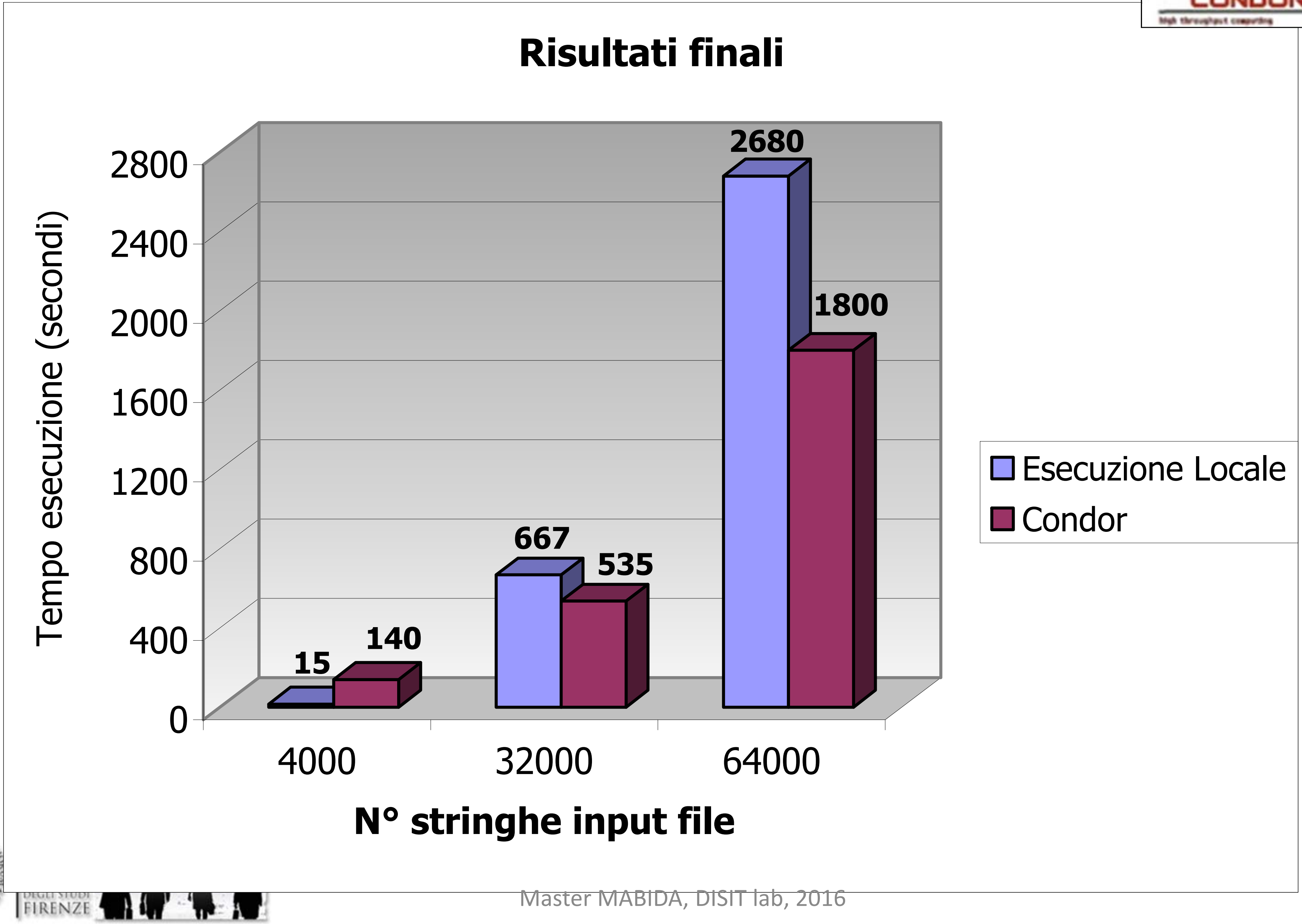
- Soluzione A)
  - analizzo 64 milioni di info su un solo calcolatore ?
- Soluzione B)
  - analizzo 64 milioni di info usando 8 calcolatori ?

- Quale delle due soluzioni ci mette meno tempo?
- E’ un problema di big data ?

An example


64 M
analisi
analisi
analisi
analisi
analisi
analisi
analisi
analisi
merge
merge
merge
merge
merge
merge
merge
File analizzato
CPU
CPU
vett ordinati
O(N + M)
P1
P2
P3
P4

Un Esperimento CONDOR at DISIT



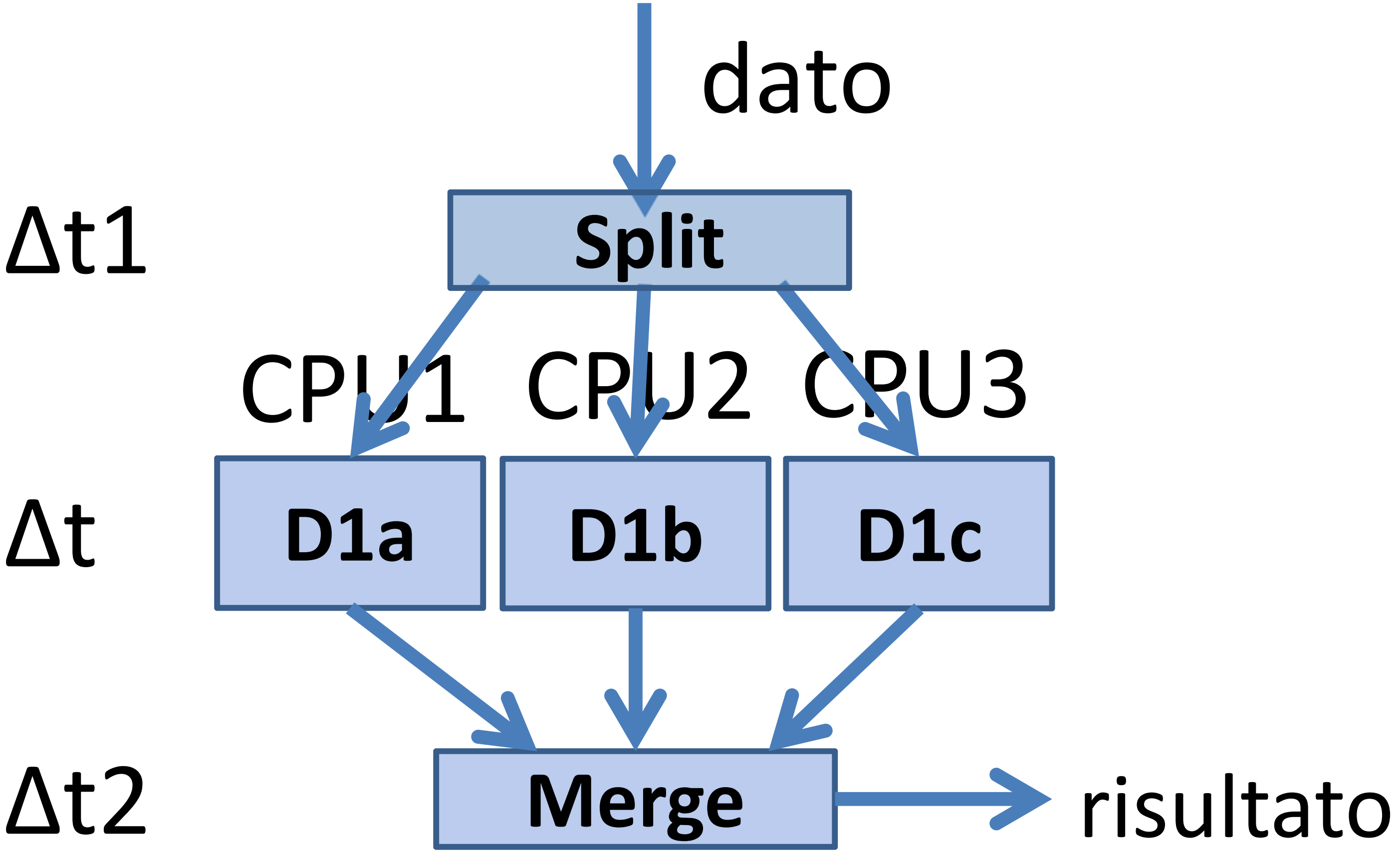
Scelte che hanno condizionato il risultato

- Si poteva utilizzare:
  - Algoritmi di analisi diversi
  - Una partizione diversa dei dati, non 8 processi ma per esempio 4, con due livelli e non 3, etc.. poteva permettere di avere maggiori vantaggi in certe condizioni
  - → spostamento del punto di ***break eaven***
- Dipendentemente dal tipo di analisi questo modello puo' non essere valido:
  - Per esempio per una convergenza

Diversi tipi di Parallelismo

- Parallelismo nei Dati:
  - Lo stesso calcolo su dati diversi
  - E.g.: trasformazione di 10 database usando lo stesso algoritmo su tutti, i dati sono disaccoppiati, si puo' fare su 10 calcolatori diversi
- Parallelismo nei processi
  - Il dato passa fasi diverse, una pipeline di produzione, con algoritmi diversi
  - Le fasi che abbiamo visto nell'esempio precedente

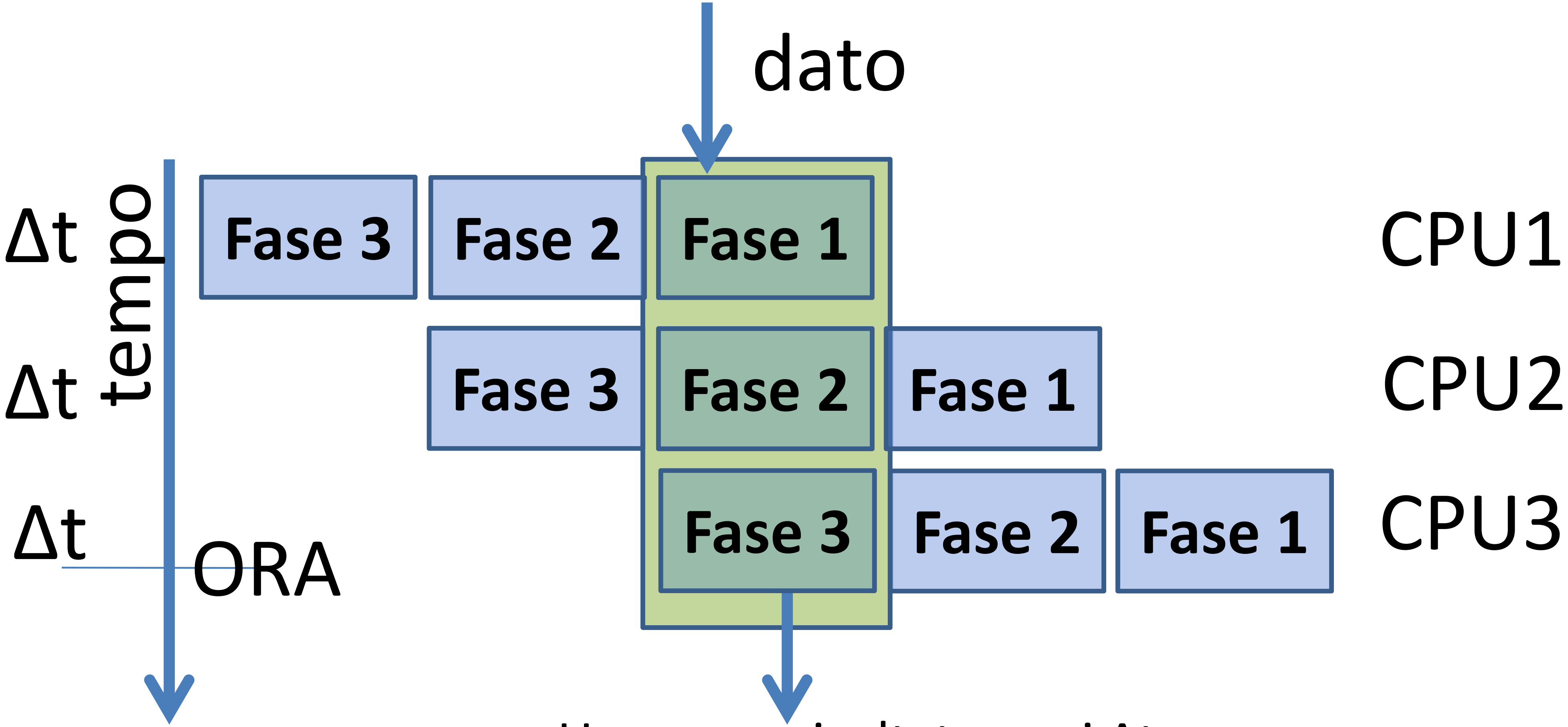
Parallelismo sul dato



Un nuovo risultato ogni T=Δt+Δt1+Δt2

Un calcolo completo in ritardo di T < 3 Δt (forse)

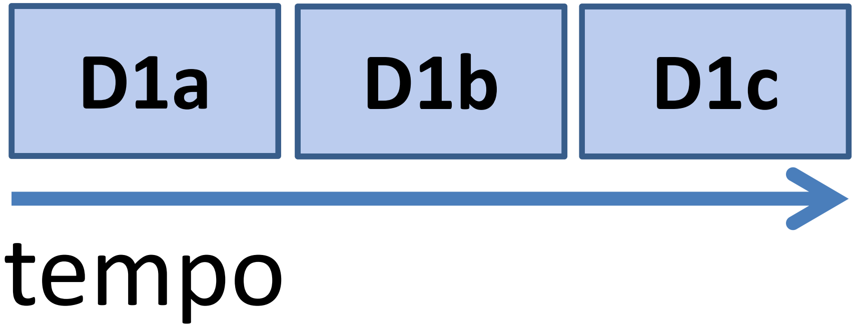
La pipeline di produzione



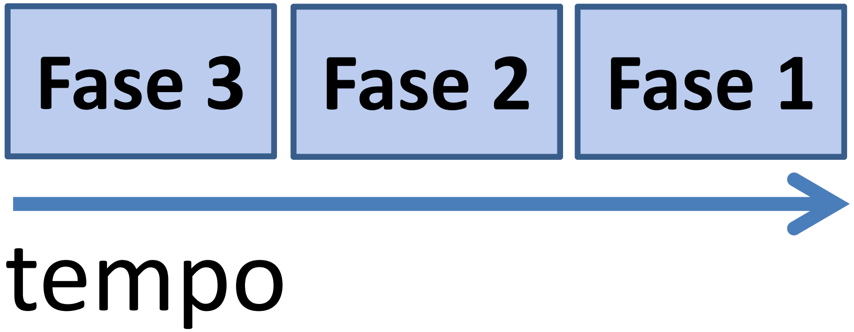
Un nuovo risultato ogni Δt

Un calcolo completo ogni 3Δt, ritardo 3Δt

Esecuzione Sequenziale



Un nuovo risultato ogni 3Δt
Un calcolo completo ogni 3Δt, ritardo 3Δt



Un nuovo risultato ogni 3Δt
Un calcolo completo ogni 3Δt, ritardo 3Δt

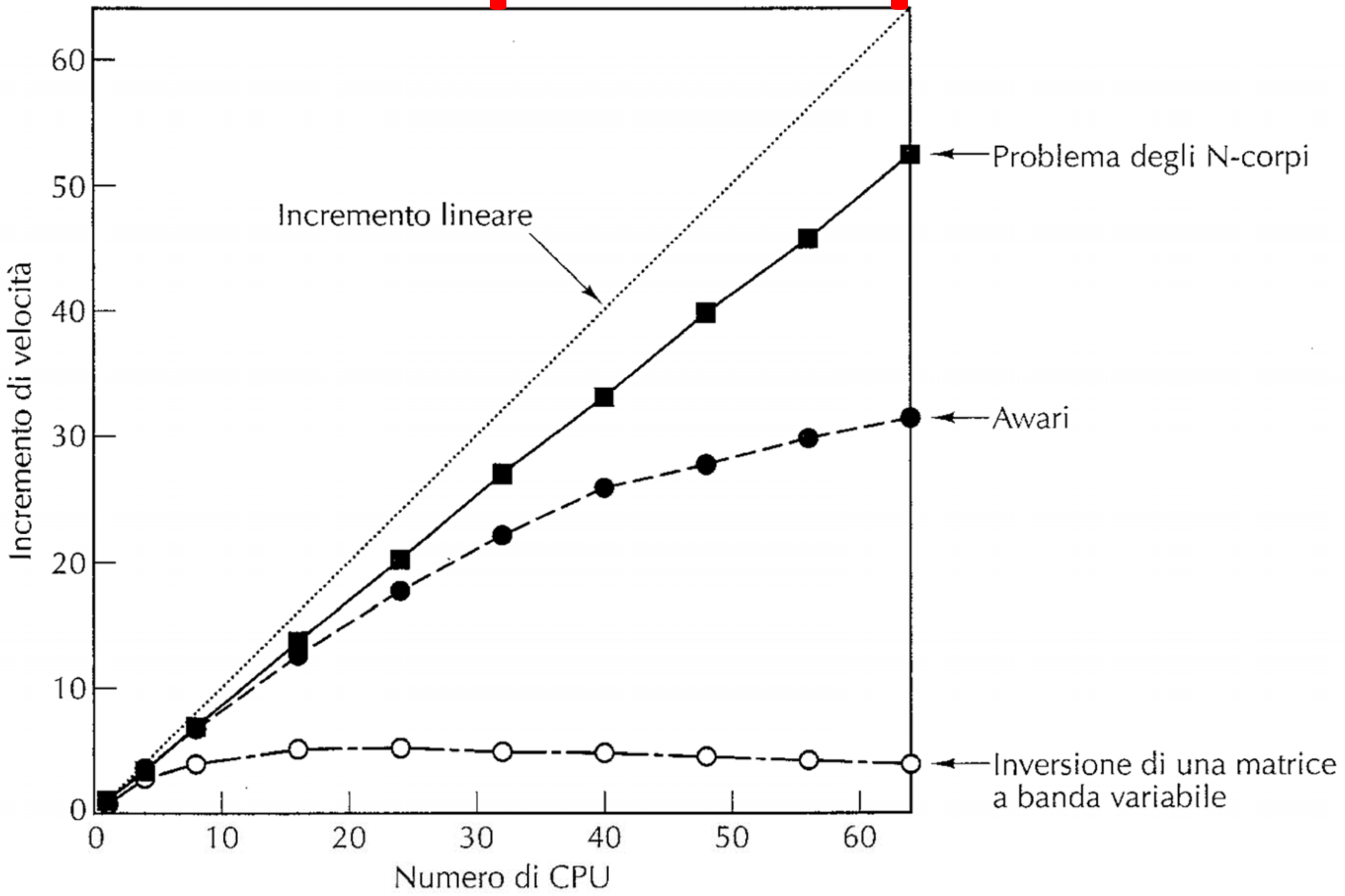
I sistemi a microprocessore moderni

- Hanno internamente capacità di esecuzione di
  - istruzioni identiche su dati diversi che
  - Istruzioni diverse su dati identici
- Hanno una pipeline di esecuzione come sistemi multi CPU Core dentro lo stessa CPU fisica
- Vi sono inoltre calcolatori che hanno più CPU fisiche che più CPU Core, etc. etc.
- Tutto questo parallelismo non e' detto che sia sfruttato al massimo dai programmi e dagli algoritmi che vengono eseguiti dai programmi commerciali a Vostra disposizione.

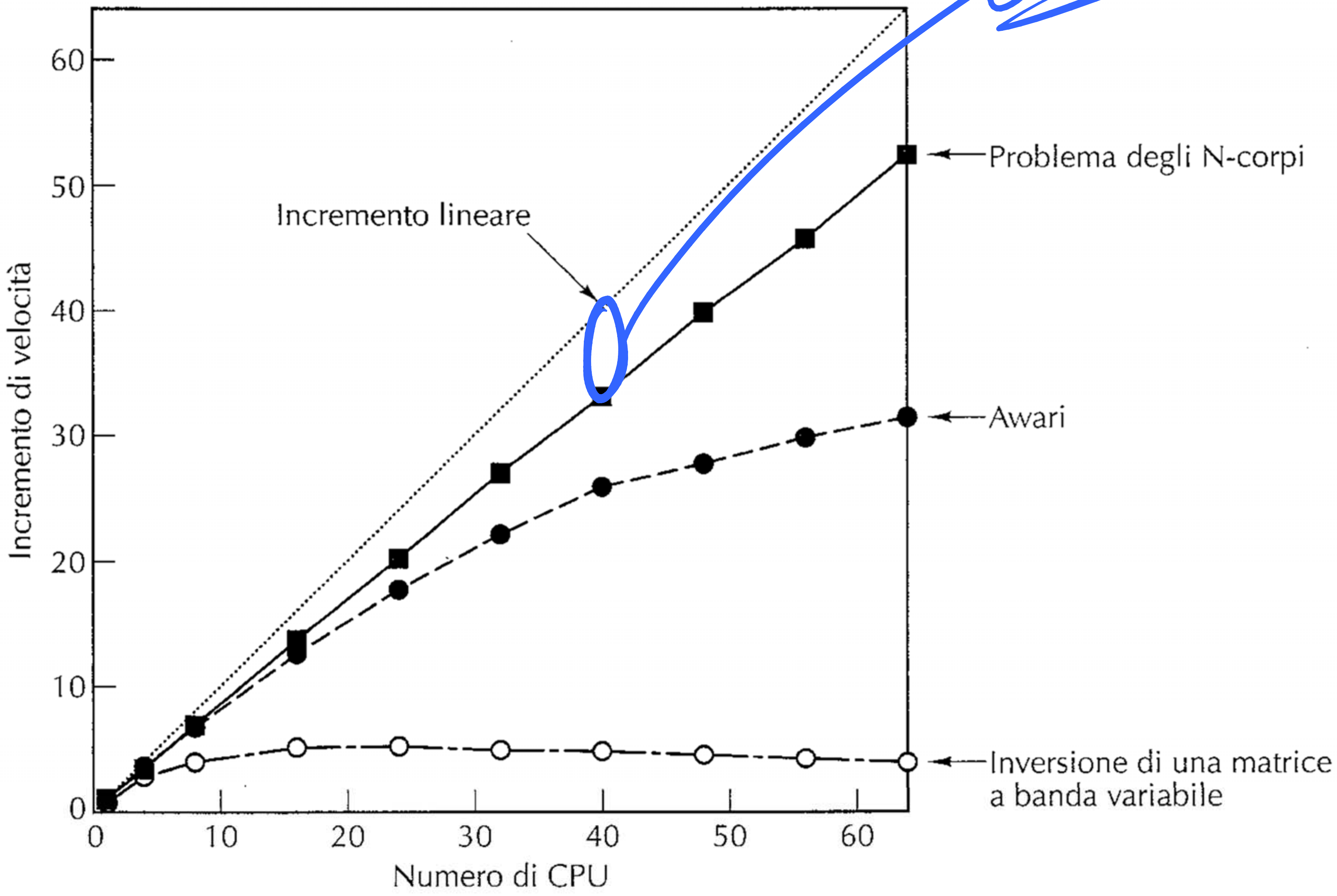
overhead

- Nel caso lineare Costo computazionale
  - O(n)
- Nel caso parallelo
  - Comunico Ai e Bi: O(n)+O(n)
  - Calcolo Ci: O(1)
  - Comunico Ci: O(n)
- Quale delle due soluzioni e’ piu’ efficiente?
  - Dipende dai dati, da N, dal costo della comunicazione, etc.
- Architetture specifiche (composizioni di processori con connessioni dedicate) possono produrre soluzioni efficaci per problemi specifici.

Speed Up


Incremento di velocità
60
50
40
30
20
10
0
0
10
20
30
40
50
60
Numero di CPU
Incremento lineare
Problema degli N-corpi
Awari
Inversione di una matrice a banda variabile

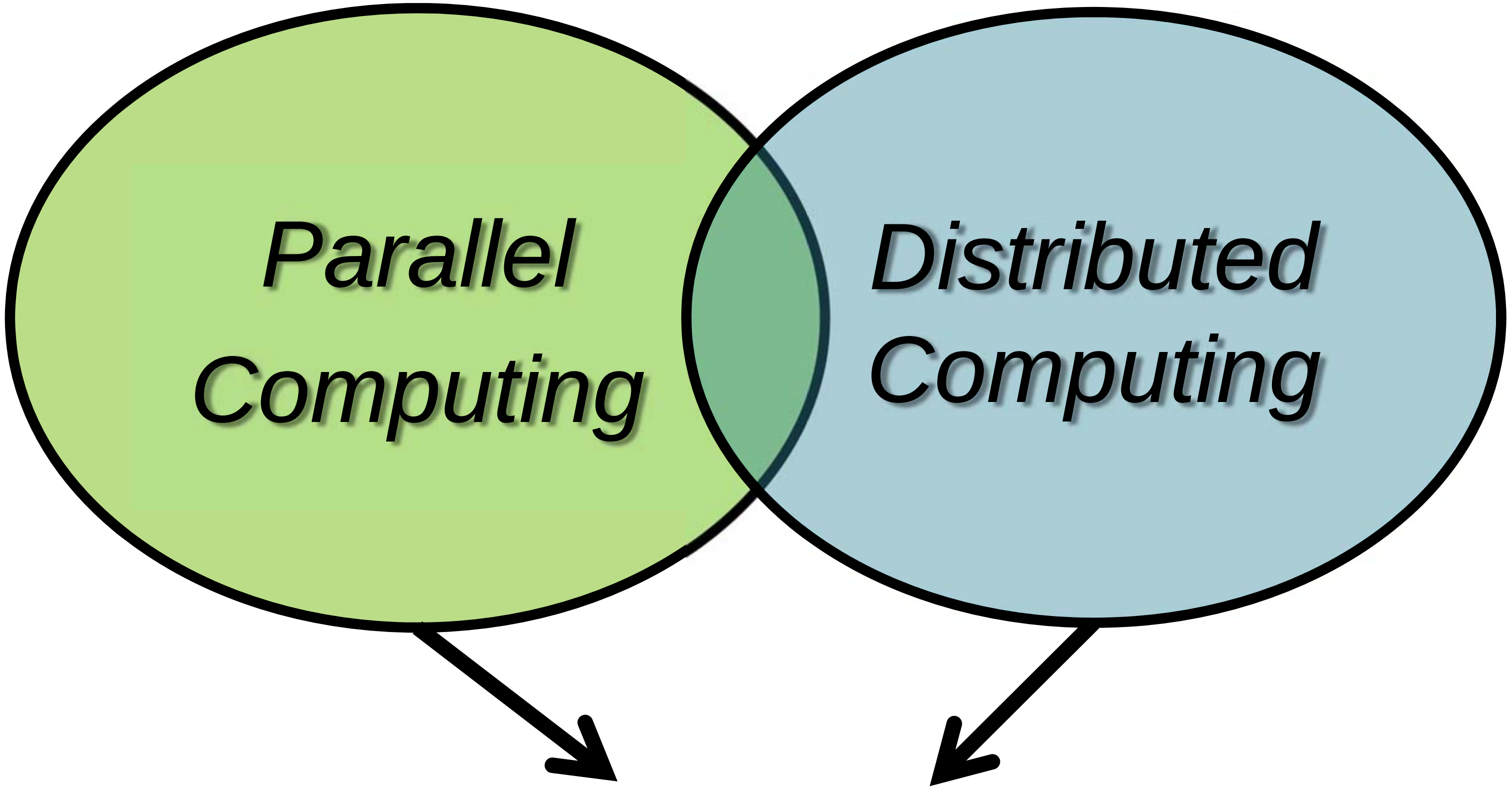
Speed Up


overhead
Incremento lineare
Problema degli N-corpi
Awari
Inversione di una matrice a banda variabile
Incremento di velocità
Numero di CPU
0
10
20
30
40
50
60

Agenda

- Big Data problems
- Big Data Pipeline of working
- Distributed and Parallel Architecture
- The grid for distributed computing
- Map Reduce Approach
- CAP Theorem
- Considerations

Grid vs Distributed and Parallel



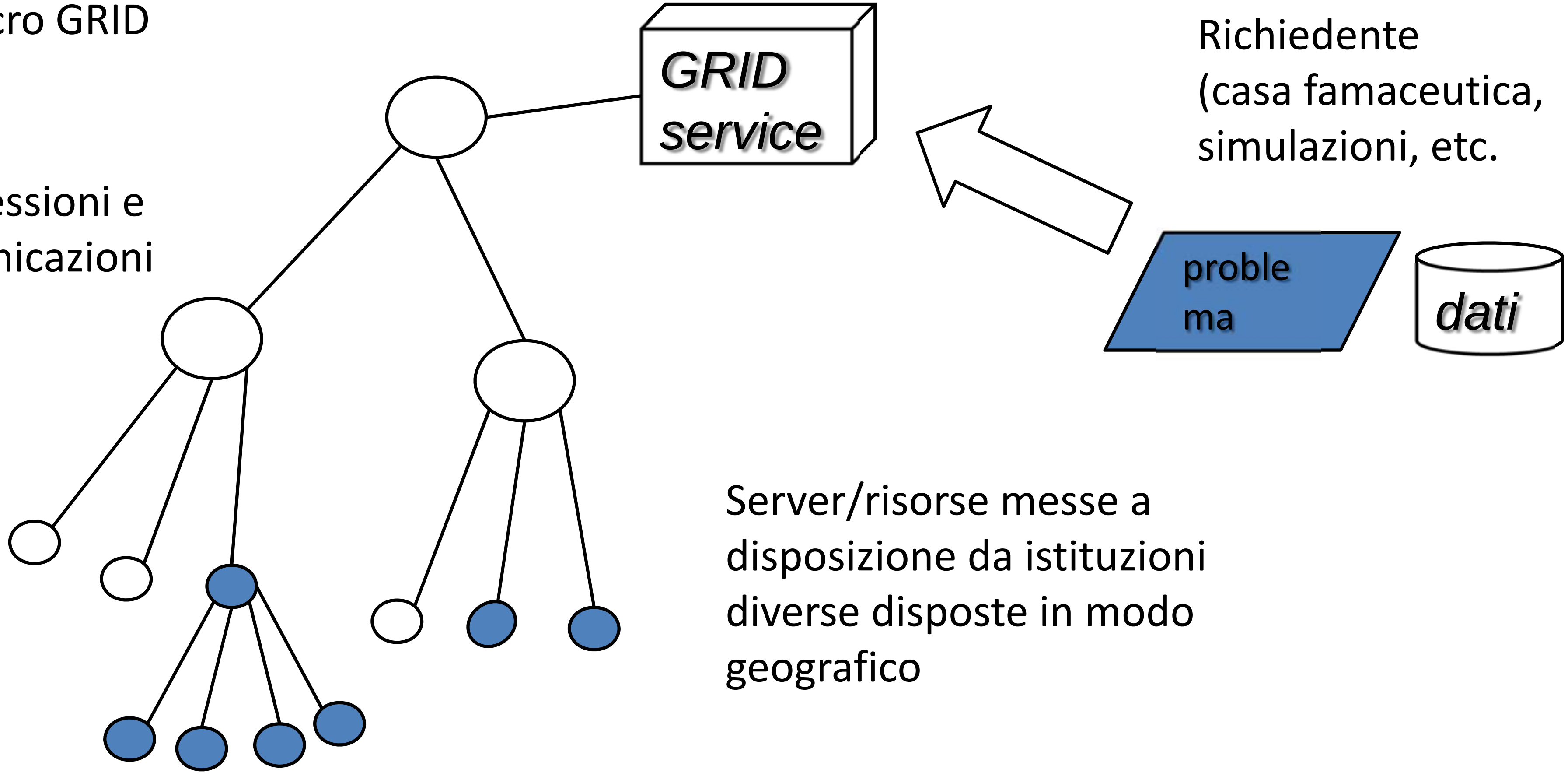
GRID Computing

The GRID

- "the **Grid**" term coined in the mid 1990s to denote a distributed computing infrastructure for advanced science and engineering
- "Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations" (Ian Foster, Karl Kesselman)

- Un insieme di risorse computazionali, di dati e reti appartenenti a diversi domini amministrativi
- Fornisce informazioni circa lo stato delle sue componenti tramite Information Services attivi e distribuiti.
- Permette agli utenti certificati di accedere alle risorse tramite un'unica procedura di autenticazione
- Gestisce gli accessi concorrenti alle risorse (compresi i fault)
  - No single point of failure

Macro GRID



Richiedente
(casa famaceutica,
simulazioni, etc.

Connessioni e
comunicazioni

Server/risorse messe a disposizione da istituzioni diverse disposte in modo geografico

Micro GRID

Per essere un GRID

- coordina risorse e fornisce meccanismi di sicurezza, policy, membership…
- Usa protocolli ed interfacce standard, open e general-purpose.
- permette l'utilizzo delle sue risorse con diversi livelli di Qualities of Service (tempo di risposta, throughput, availability, sicurezza…).
- L'utilità del sistema (middle tier) e molto maggiore a quella della somma delle sue parti nel supporto alle necessità dell'utente.

Some GRID Solutions !!

- **Condor**
  - Unix and windows
  - Small scale GRID, non parallelism
- **Globus**
  - Parallel
  - Unix like
  - C and java
- **Legion**
  - Parallel, C++
  - Unix like
  - Too much space needed, 300Mbyte
- **Unicore**
  - Java
  - Unix like
  - Open source
- **AXMEDIS, AXCP**
  - C++ and JavaScript
  - Windows
  - Accessible Code, Free Software

Scienze Data Intensive

- Fisica nucleare e delle alte energie
  - Nuovi esperimenti del CERN
- Ricerca onde gravitazionali
  - LIGO, GEO, VIRGO
- Analisi di serie temporali di dati 3D (simulazione, osservazione)
  - Earth Observation, Studio del clima
  - Geofisica, Previsione dei terremoti
  - Fluido, Aerodinamica
  - Diffusione inquinanti
- Astronomia: Digital sky surveys

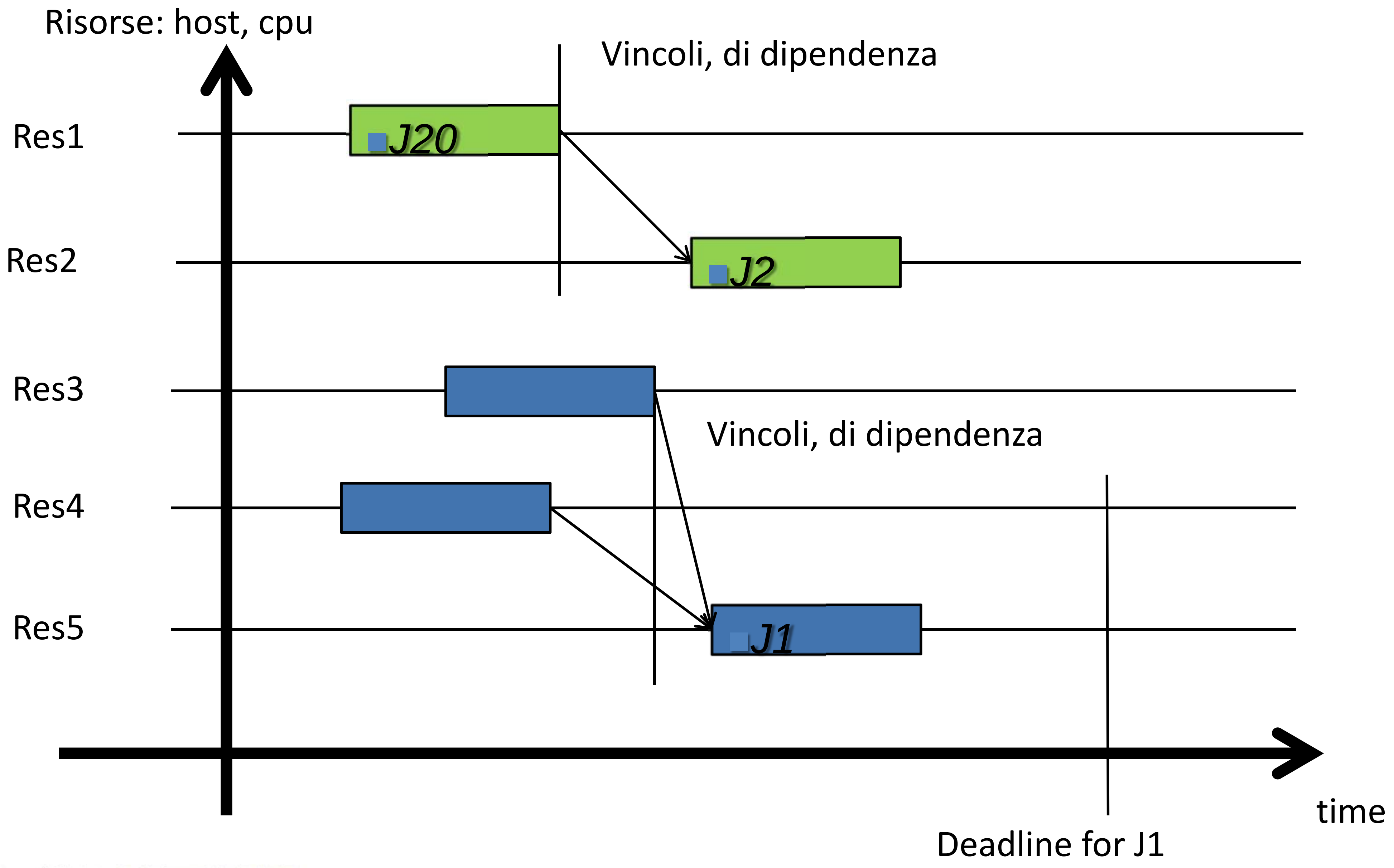
BigData Science

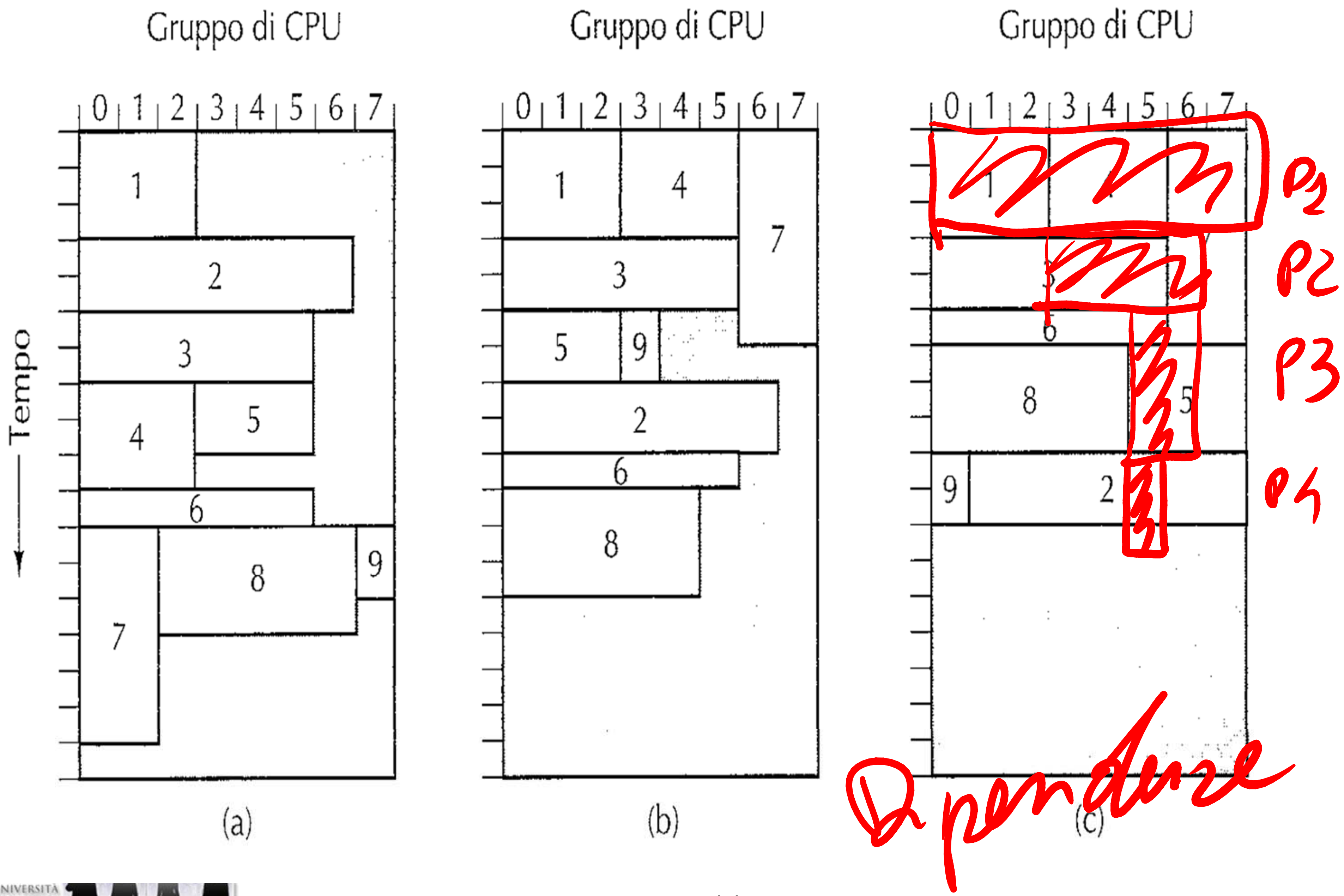
- How to: compute, process, simulate, extract meaning, of vaste/huge amount of data to create knowledge
- Exploit efficiently the hugh amount of data/knowledge
- Issues:
  - Data representation: indexing, search, execute, etc..
  - Data computing: sparse, uncertain, fast, etc.
  - Data understanding: mining, analize, etc.
  - Data view: fast, navigate,
- Applications:
  - Recommendations, suggestions, semantic computing
  - Business intelligence: health, trends, genomic, HBP,
  - Distributed database, decision taking
  - Social networking, smart city
  - Event detection, unexpected correlation

Scheduling Distribuito

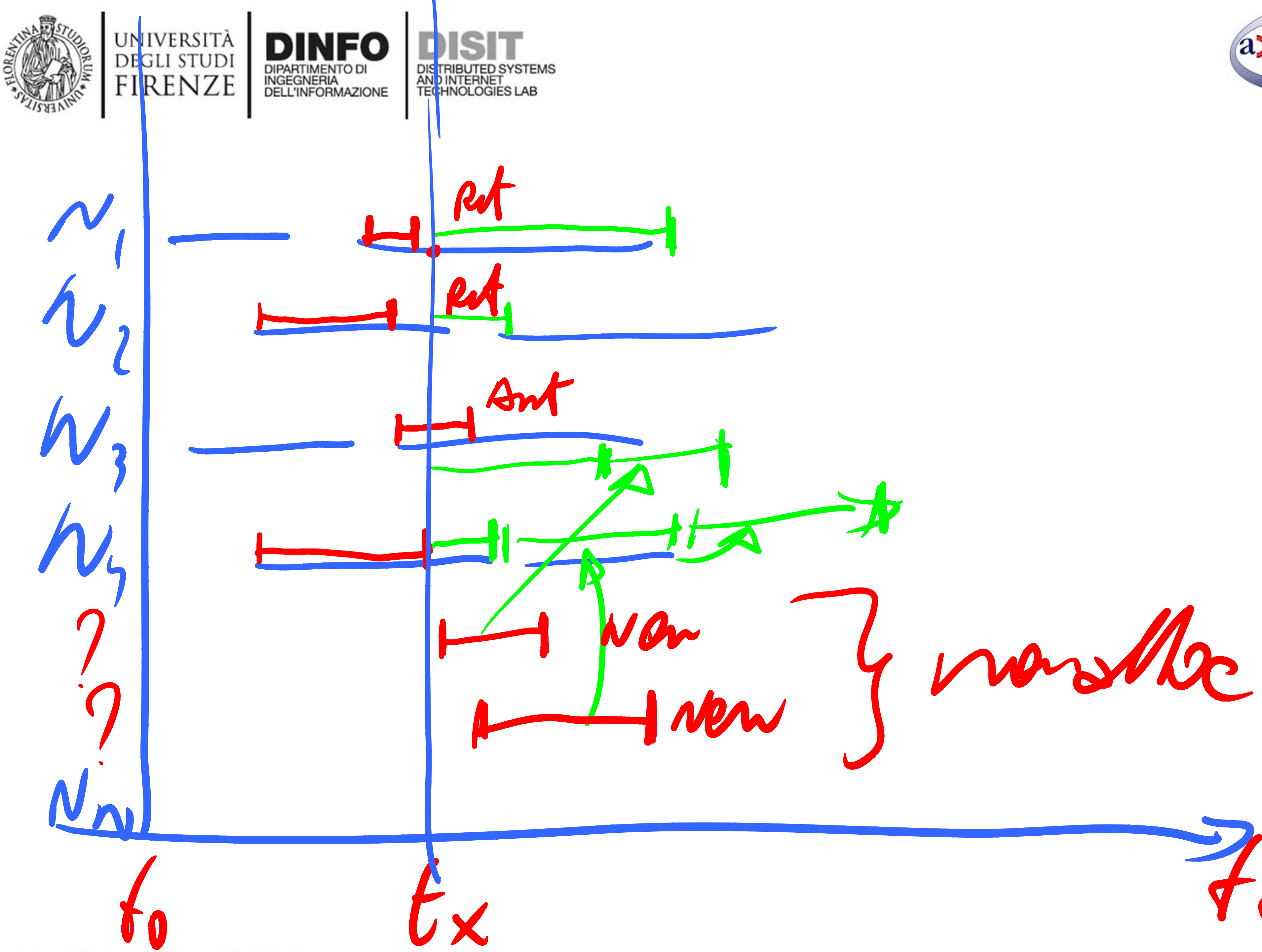
- N nodi (calcolatori, VM) che possono eseguire processi diversi: valido per:
  - Caso A) Parallelismo sui dati
  - Caso B) Parallelismo sui processi
- Un Server che come scheduler gestisce i processi, gli alloca sui nodi
  - Al momento giusto
  - Considerando le loro dipendenze
  - Con modelli predittivi sulla loro durata
  - Adattando nel tempo l'allocazione
  - Etc.

Gantt diagram and jobs




Gruppo di CPU
Gruppo di CPU
Gruppo di CPU
0 1 2 3 4 5 6 7
Tempo
1
2
3
4
5
6
7
8
9
1
4
7
3
5
9
2
6
8
1
4
3
6
8
5
9
2
P1
P2
P3
P4
Dipendenze

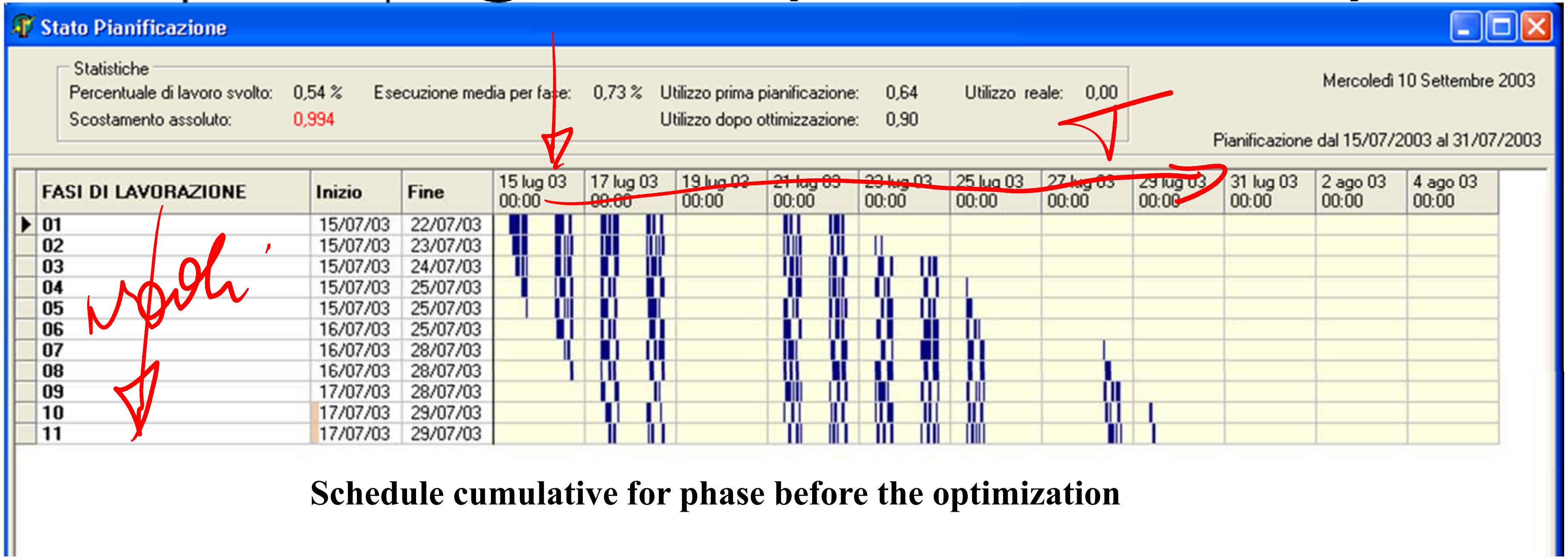
(a) (b) (c)


N1
N2
N3
N4
?
?
Nn
Rel
Rel
Ant
new
new
t0
tx
t0

Per ogni processo al tempo Tx

- D: Durata prevista del processo (con incertezza)
- Tstart: Tempo di start (time stamp di start)
- Tend: Tempo previsto di completamento
- Pc: Percentuale di completamento (se possibile), per esempio
  - Valore dell'indice di un processo iterativo,
  - numero di iterazioni, etc. etc.
- Ad un certo Tx>Tstart la Pc puo' essere:
  - Conforme (Tx-Tstart)/D%==Pc
  - In anticipo Pc > (Tx-Tstart)/D%
  - In ritardo   Pc < (Tx-Tstart)/D%
- Necessario attualizzare

Gantt Diagrams (OPTAMS, TS)



Stato Pianificazione

Statistiche

Percentuale di lavoro svolto: 0,54 % Esecuzione media per fase: 0,73 % Utilizzo prima pianificazione: 0,64 Utilizzo reale: 0,00

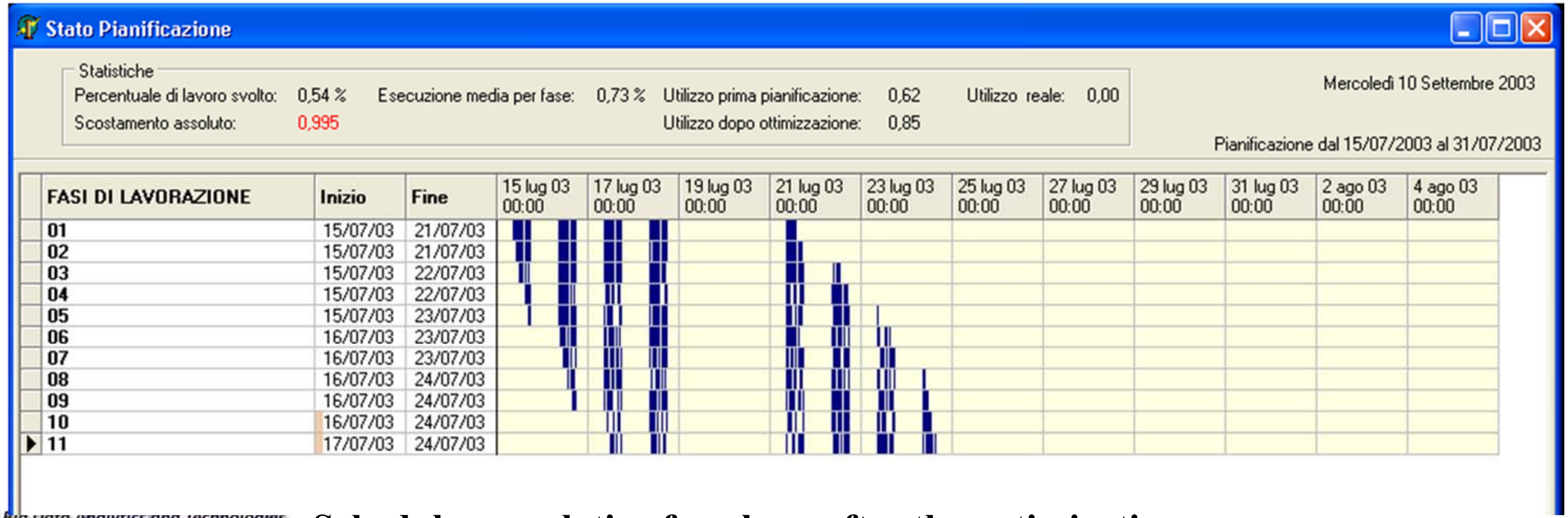
Scostamento assoluto: 0,994 Utilizzo dopo ottimizzazione: 0,90

Mercoledì 10 Settembre 2003

Pianificazione dal 15/07/2003 al 31/07/2003

FASI DI LAVORAZIONE	Inizio	Fine
01	15/07/03	22/07/03
02	15/07/03	23/07/03
03	15/07/03	24/07/03
04	15/07/03	25/07/03
05	15/07/03	25/07/03
06	16/07/03	25/07/03
07	16/07/03	28/07/03
08	16/07/03	28/07/03
09	17/07/03	28/07/03
10	17/07/03	29/07/03
11	17/07/03	29/07/03

Schedule cumulative for phase before the optimization



Stato Pianificazione

Statistiche

Percentuale di lavoro svolto: 0,54 % Esecuzione media per fase: 0,73 % Utilizzo prima pianificazione: 0,62 Utilizzo reale: 0,00

Scostamento assoluto: 0,995 Utilizzo dopo ottimizzazione: 0,85

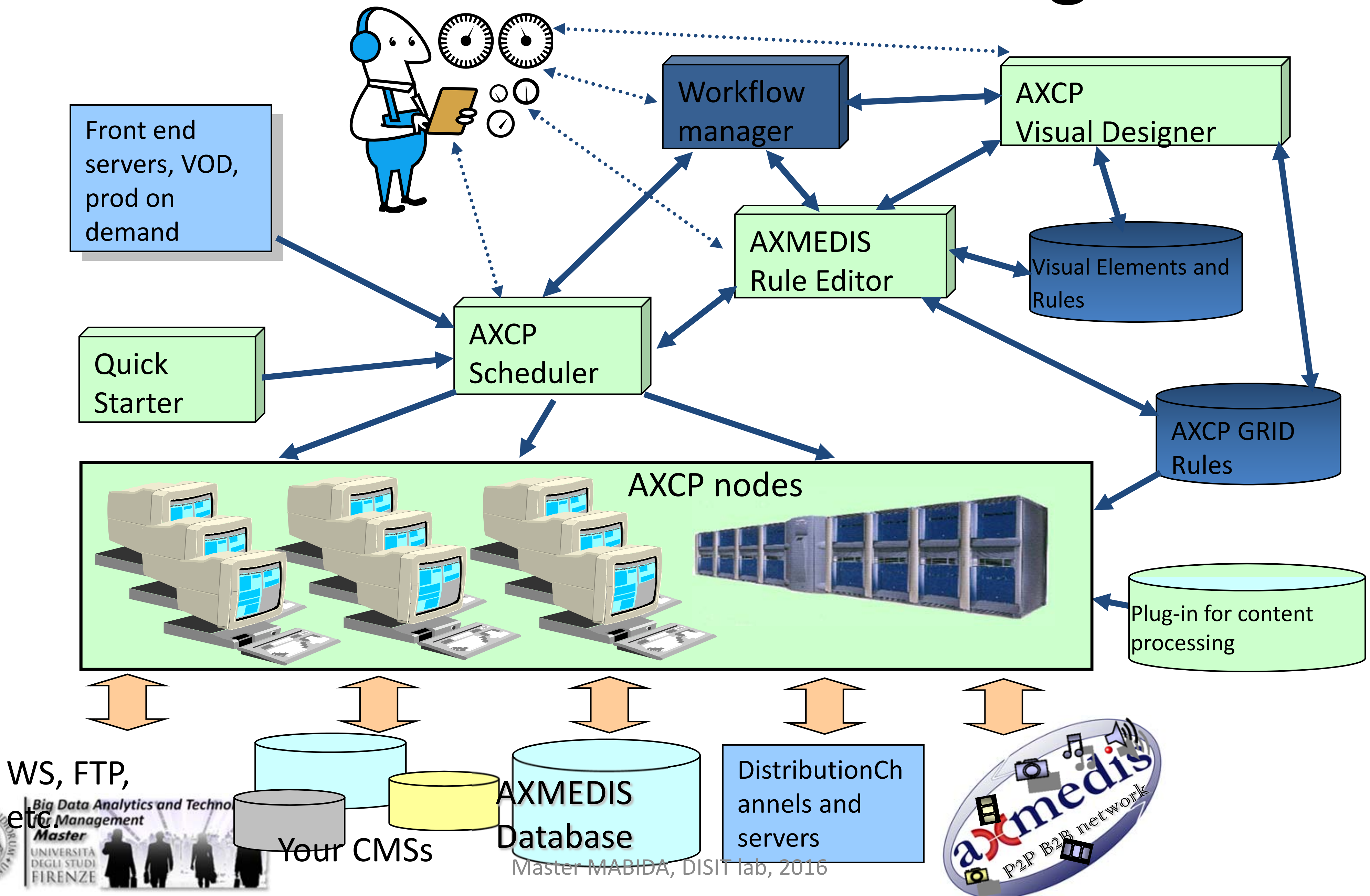
Mercoledì 10 Settembre 2003

Pianificazione dal 15/07/2003 al 31/07/2003

FASI DI LAVORAZIONE	Inizio	Fine
01	15/07/03	21/07/03
02	15/07/03	21/07/03
03	15/07/03	22/07/03
04	15/07/03	22/07/03
05	15/07/03	23/07/03
06	16/07/03	23/07/03
07	16/07/03	23/07/03
08	16/07/03	24/07/03
09	16/07/03	24/07/03
10	16/07/03	24/07/03
11	17/07/03	24/07/03

Schedule cumulative for phase after the optimization

AXMEDIS Content Processing GRID



Workflow manager

AXCP Visual Designer

Front end servers, VOD, prod on demand

AXMEDIS Rule Editor

Visual Elements and Rules

AXCP Scheduler

Quick Starter

AXCP GRID Rules

AXCP nodes

Plug-in for content processing

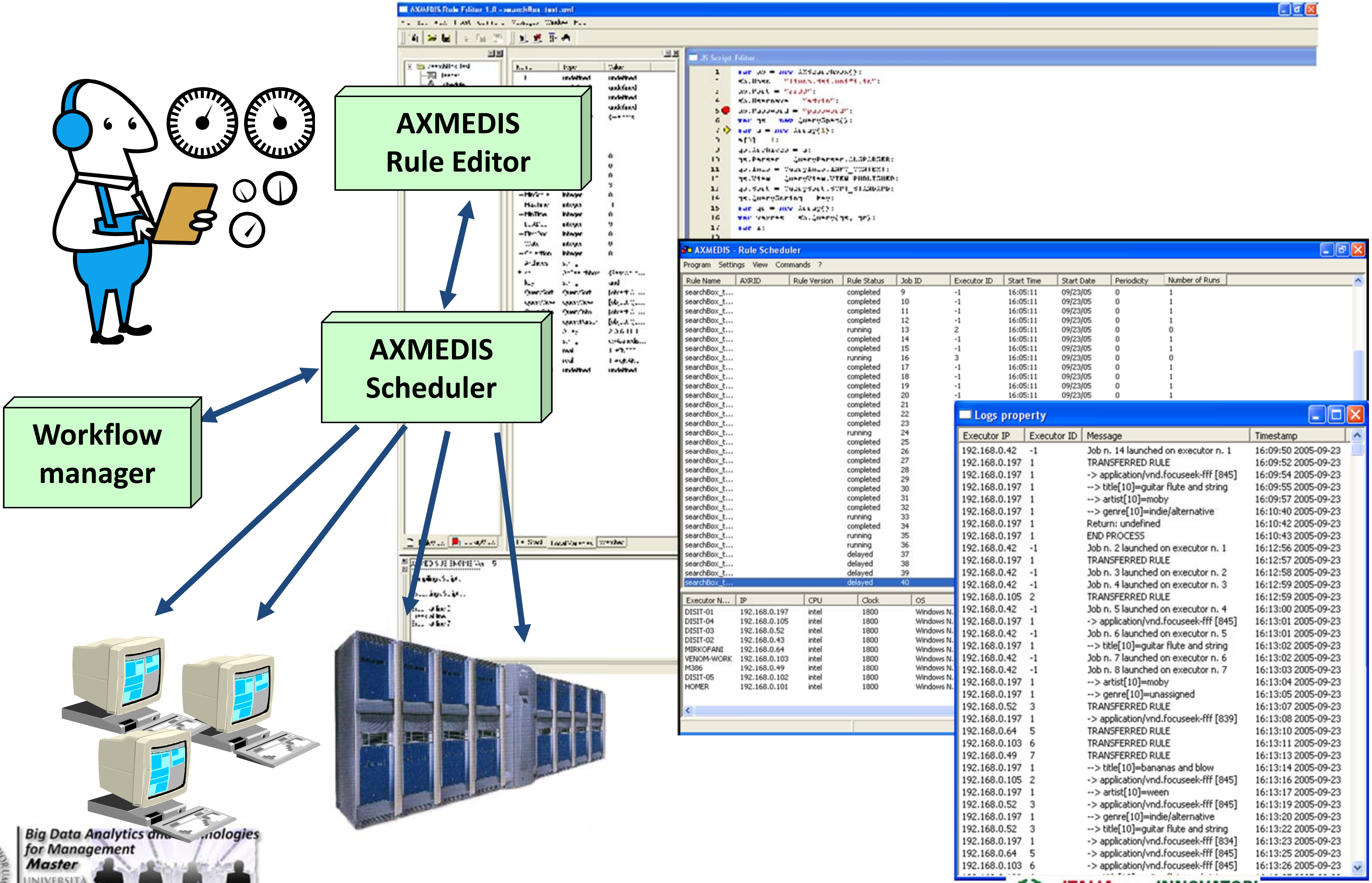
WS, FTP, etc.

Your CMSs

AXMEDIS Database

DistributionChannels and servers

AXMEDIS Content Processing GRID



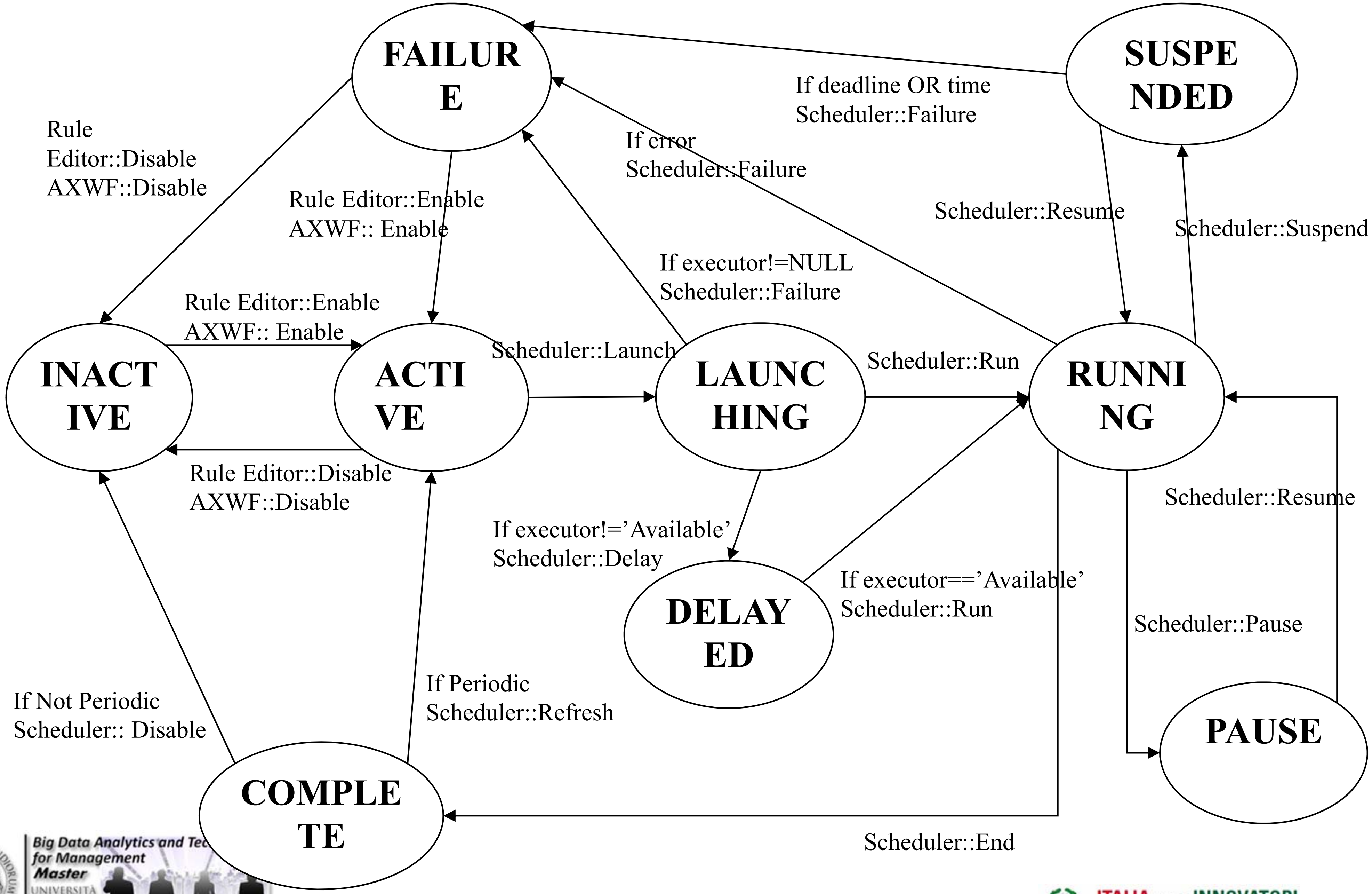
Rule Name	AXRID	Rule Version	Rule Status	Job ID	Executor ID	Start Time	Start Date	Periodicity	Number of Runs
searchBox_t...			completed	9	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	10	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	11	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	12	-1	16:05:11	09/23/05	0	1
searchBox_t...			running	13	2	16:05:11	09/23/05	0	0
searchBox_t...			completed	14	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	15	-1	16:05:11	09/23/05	0	1
searchBox_t...			running	16	3	16:05:11	09/23/05	0	0
searchBox_t...			completed	17	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	18	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	19	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	20	-1	16:05:11	09/23/05	0	1
searchBox_t...			completed	21					
searchBox_t...			completed	22					
searchBox_t...			completed	23					
searchBox_t...			running	24					
searchBox_t...			completed	25					
searchBox_t...			completed	26					
searchBox_t...			completed	27					
searchBox_t...			completed	28					
searchBox_t...			completed	29					
searchBox_t...			completed	30					
searchBox_t...			completed	31					
searchBox_t...			completed	32					
searchBox_t...			running	33					
searchBox_t...			completed	34					
searchBox_t...			running	35					
searchBox_t...			running	36					
searchBox_t...			delayed	37					
searchBox_t...			delayed	38					
searchBox_t...			delayed	39					
searchBox_t...			delayed	40					

Executor N...	IP	CPU	Clock	OS
DISIT-01	192.168.0.197	intel	1800	Windows N.
DISIT-04	192.168.0.105	intel	1800	Windows N.
DISIT-03	192.168.0.52	intel	1800	Windows N.
DISIT-02	192.168.0.43	intel	1800	Windows N.
MIRKOFANI	192.168.0.64	intel	1800	Windows N.
VENOM-WORK	192.168.0.103	intel	1800	Windows N.
M386	192.168.0.49	intel	1800	Windows N.
DISIT-05	192.168.0.102	intel	1800	Windows N.
HOMER	192.168.0.101	intel	1800	Windows N.

Executor IP	Executor ID	Message	Timestamp
192.168.0.42	-1	Job n. 14 launched on executor n. 1	16:09:50 2005-09-23
192.168.0.197	1	TRANSFERRED RULE	16:09:52 2005-09-23
192.168.0.197	1	-> application/vnd.focuseek-fff [845]	16:09:54 2005-09-23
192.168.0.197	1	--> title[10]=guitar flute and string	16:09:55 2005-09-23
192.168.0.197	1	--> artist[10]=moby	16:09:57 2005-09-23
192.168.0.197	1	--> genre[10]=indie/alternative	16:10:40 2005-09-23
192.168.0.197	1	Return: undefined	16:10:42 2005-09-23
192.168.0.197	1	END PROCESS	16:10:43 2005-09-23
192.168.0.42	-1	Job n. 2 launched on executor n. 1	16:12:56 2005-09-23
192.168.0.197	1	TRANSFERRED RULE	16:12:57 2005-09-23
192.168.0.42	-1	Job n. 3 launched on executor n. 2	16:12:58 2005-09-23
192.168.0.42	-1	Job n. 4 launched on executor n. 3	16:12:59 2005-09-23
192.168.0.105	2	TRANSFERRED RULE	16:12:59 2005-09-23
192.168.0.42	-1	Job n. 5 launched on executor n. 4	16:13:00 2005-09-23
192.168.0.197	1	-> application/vnd.focuseek-fff [845]	16:13:01 2005-09-23
192.168.0.42	-1	Job n. 6 launched on executor n. 5	16:13:01 2005-09-23
192.168.0.197	1	--> title[10]=guitar flute and string	16:13:02 2005-09-23
192.168.0.42	-1	Job n. 7 launched on executor n. 6	16:13:02 2005-09-23
192.168.0.42	-1	Job n. 8 launched on executor n. 7	16:13:03 2005-09-23
192.168.0.197	1	--> artist[10]=moby	16:13:04 2005-09-23
192.168.0.197	1	--> genre[10]=unassigned	16:13:05 2005-09-23
192.168.0.52	3	TRANSFERRED RULE	16:13:07 2005-09-23
192.168.0.197	1	-> application/vnd.focuseek-fff [839]	16:13:08 2005-09-23
192.168.0.64	5	TRANSFERRED RULE	16:13:10 2005-09-23
192.168.0.103	6	TRANSFERRED RULE	16:13:11 2005-09-23
192.168.0.49	7	TRANSFERRED RULE	16:13:13 2005-09-23
192.168.0.197	1	--> title[10]=bananas and blow	16:13:14 2005-09-23
192.168.0.105	2	-> application/vnd.focuseek-fff [845]	16:13:16 2005-09-23
192.168.0.197	1	--> artist[10]=ween	16:13:17 2005-09-23
192.168.0.52	3	-> application/vnd.focuseek-fff [845]	16:13:19 2005-09-23
192.168.0.197	1	--> genre[10]=indie/alternative	16:13:20 2005-09-23
192.168.0.52	3	--> title[10]=guitar flute and string	16:13:22 2005-09-23
192.168.0.197	1	-> application/vnd.focuseek-fff [834]	16:13:23 2005-09-23
192.168.0.64	5	-> application/vnd.focuseek-fff [845]	16:13:25 2005-09-23
192.168.0.103	6	-> application/vnd.focuseek-fff [845]	16:13:26 2005-09-23

ITALIA DEGLI INNOVATORI
Agenzia per la Diffusione delle Tecnologie per l'Innovazione
Presidenza del Consiglio dei Ministri

Evolution of Rule's State



FAILURE

SUSPENDED

INACTIVE

ACTIVE

LAUNCHING

RUNNING

DELAYED

PAUSE

COMPLETE

Rule Editor::Disable
AXWF::Disable

Rule Editor::Enable
AXWF:: Enable

Rule Editor::Enable
AXWF:: Enable

Rule Editor::Disable
AXWF::Disable

If error
Scheduler::Failure

If deadline OR time
Scheduler::Failure

If executor!=NULL
Scheduler::Failure

Scheduler::Resume

Scheduler::Suspend

Scheduler::Launch

Scheduler::Run

If executor!='Available'
Scheduler::Delay

If executor=='Available'
Scheduler::Run

Scheduler::Resume

Scheduler::Pause

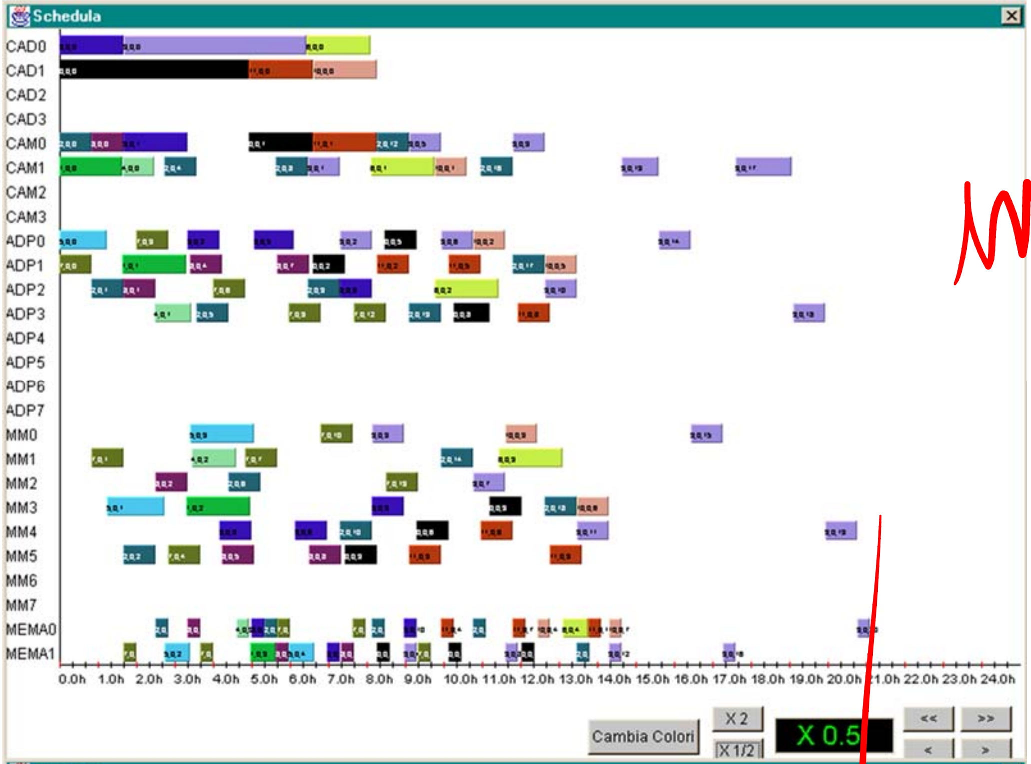
If Not Periodic
Scheduler:: Disable

If Periodic
Scheduler::Refresh

Scheduler::End

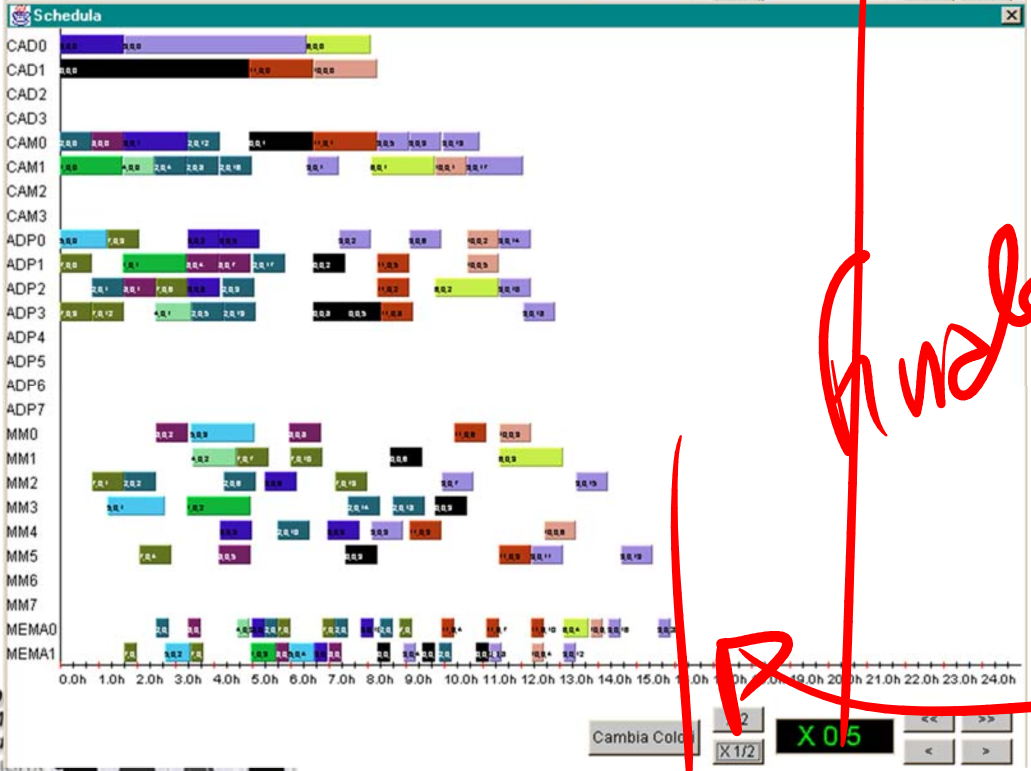
Esempio di ottimizzazione

iniziale


Schedula
CAD0
CAD1
CAD2
CAD3
CAM0
CAM1
CAM2
CAM3
ADP0
ADP1
ADP2
ADP3
ADP4
ADP5
ADP6
ADP7
MM0
MM1
MM2
MM3
MM4
MM5
MM6
MM7
MEMA0
MEMA1
0.0h 1.0h 2.0h 3.0h 4.0h 5.0h 6.0h 7.0h 8.0h 9.0h 10.0h 11.0h 12.0h 13.0h 14.0h 15.0h 16.0h 17.0h 18.0h 19.0h 20.0h 21.0h 22.0h 23.0h 24.0h
Cambia Colori
X 2
X 1/2
X 0.5
<<
>>
<
>

125 task, 2CAD, 2CAM, 4ADP, 6MM, 2MEMA

finale

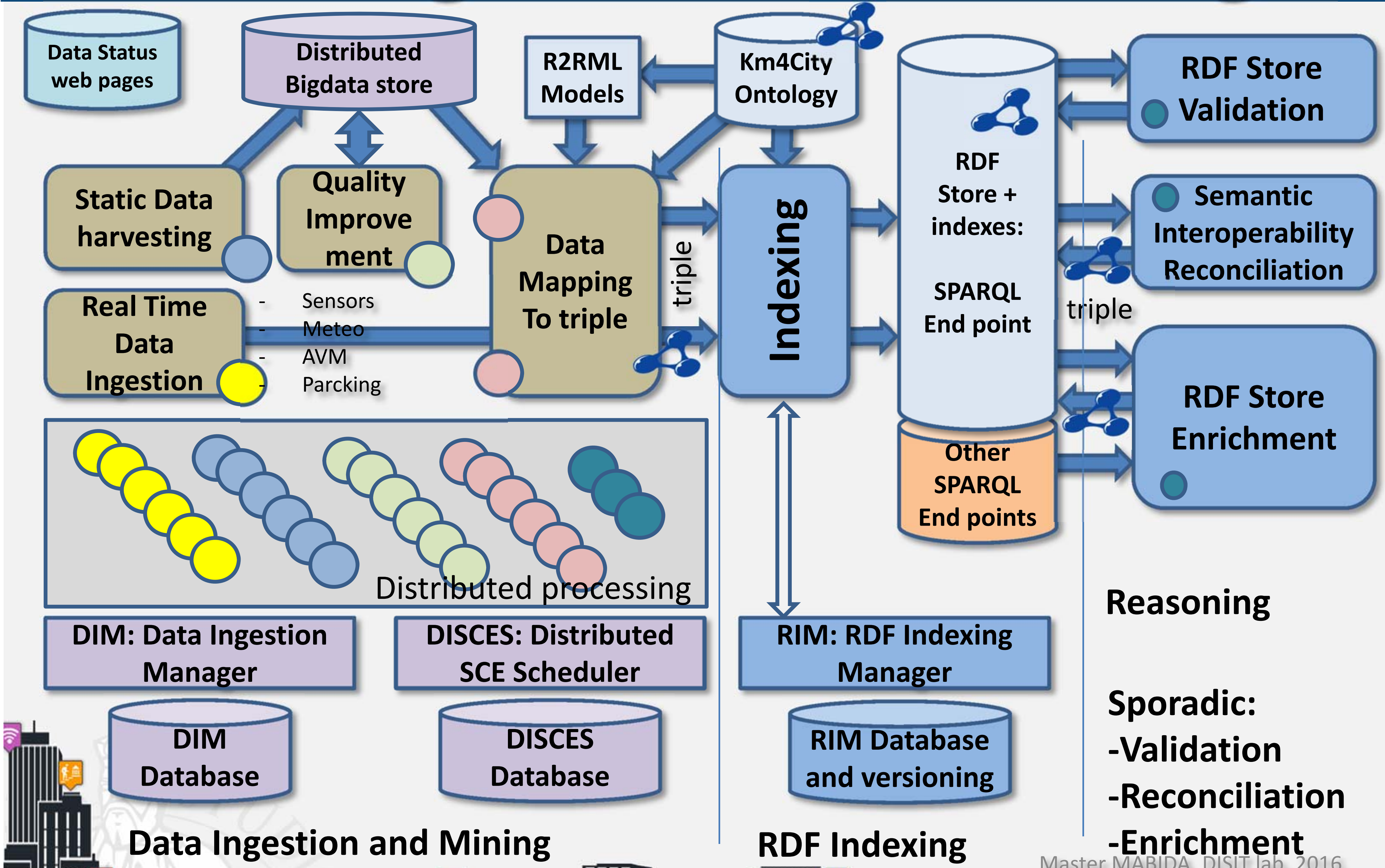

Schedula
CAD0
CAD1
CAD2
CAD3
CAM0
CAM1
CAM2
CAM3
ADP0
ADP1
ADP2
ADP3
ADP4
ADP5
ADP6
ADP7
MM0
MM1
MM2
MM3
MM4
MM5
MM6
MM7
MEMA0
MEMA1
0.0h 1.0h 2.0h 3.0h 4.0h 5.0h 6.0h 7.0h 8.0h 9.0h 10.0h 11.0h 12.0h 13.0h 14.0h 15.0h 16.0h 17.0h 18.0h 19.0h 20.0h 21.0h 22.0h 23.0h 24.0h
Cambia Colori
X 2
X 1/2
X 0.5
<<
>>
<
>

Ottimizzazione: 24.6%

DISCES: Distributed SCE System

- Architettura distribuita con scheduler distribuito che viene replicato su ogni nodo di un cluster di server
- Ogni Nodo accede al database distribuito per sapere cosa deve fare e prende in carico un lavoro di elaborazione
- I processi sono eseguibili su
  - N nodi in sequenza o in parallelo, si possono definire delle dipendenze
  - In modo periodico o aperiodico (on demand)
- Al momento utilizzato su varie soluzioni big data:
  - Smart city Firenze, smart cloud, simulazione, social network, etc.

Data Ingestion and Mining



Data Status web pages

Distributed Bigdata store

R2RML Models

Km4City Ontology

Static Data harvesting

Quality Improve ment

Data Mapping To triple

triple

Indexing

RDF Store + indexes:

SPARQL End point

Other SPARQL End points

RDF Store Validation

Semantic Interoperability Reconciliation

triple

RDF Store Enrichment

Real Time Data Ingestion

- Sensors
- Meteo
- AVM
- Parcking

Distributed processing

DIM: Data Ingestion Manager

DISCES: Distributed SCE Scheduler

RIM: RDF Indexing Manager

DIM Database

DISCES Database

RIM Database and versioning

Data Ingestion and Mining

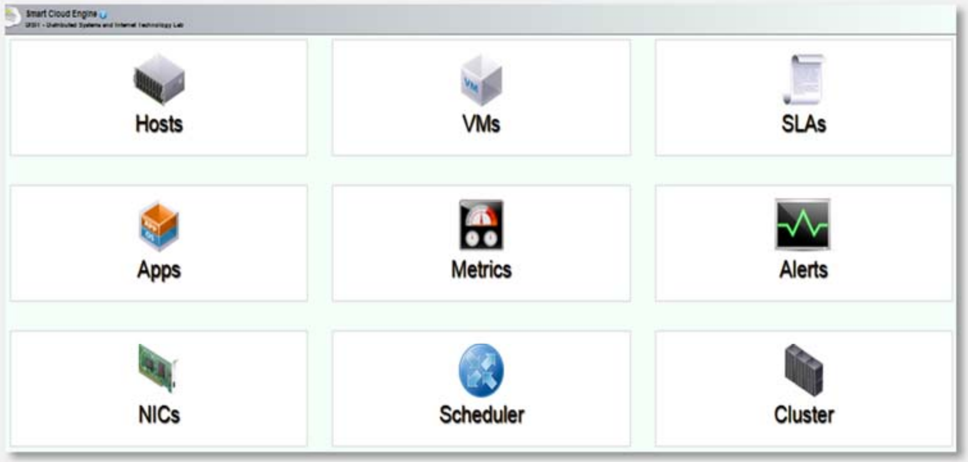
RDF Indexing

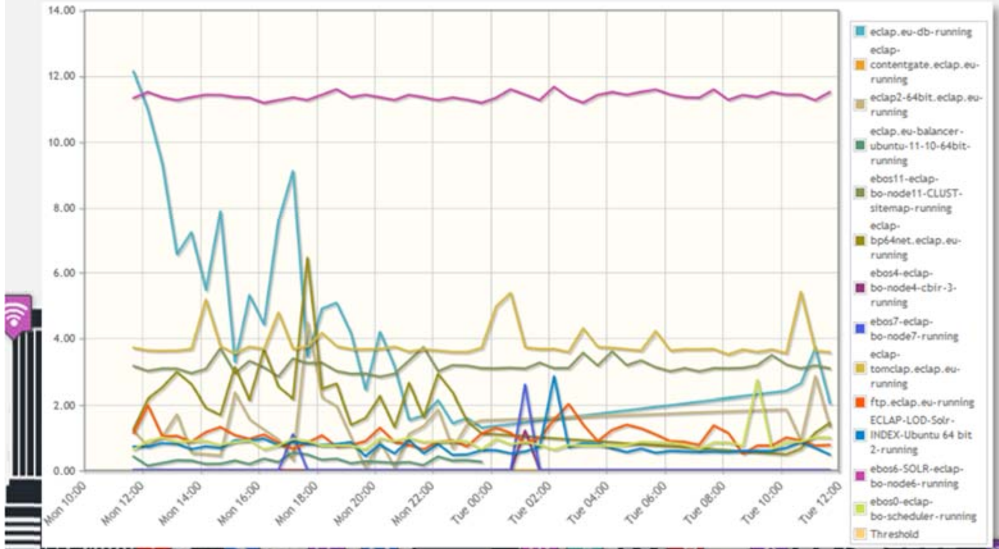
Reasoning

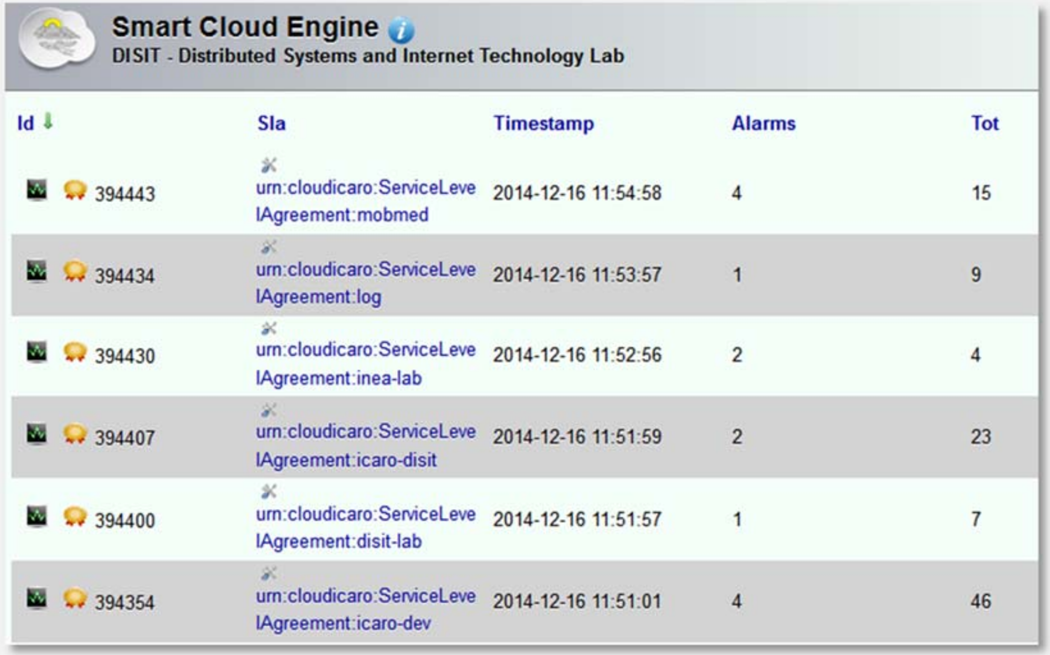
Sporadic:
-Validation
-Reconciliation
-Enrichment

Smart City Engine

- Status: in progress
- SLA derived from Cloud SLA
- Model mutuated from Smart Cloud Engine of ICARO



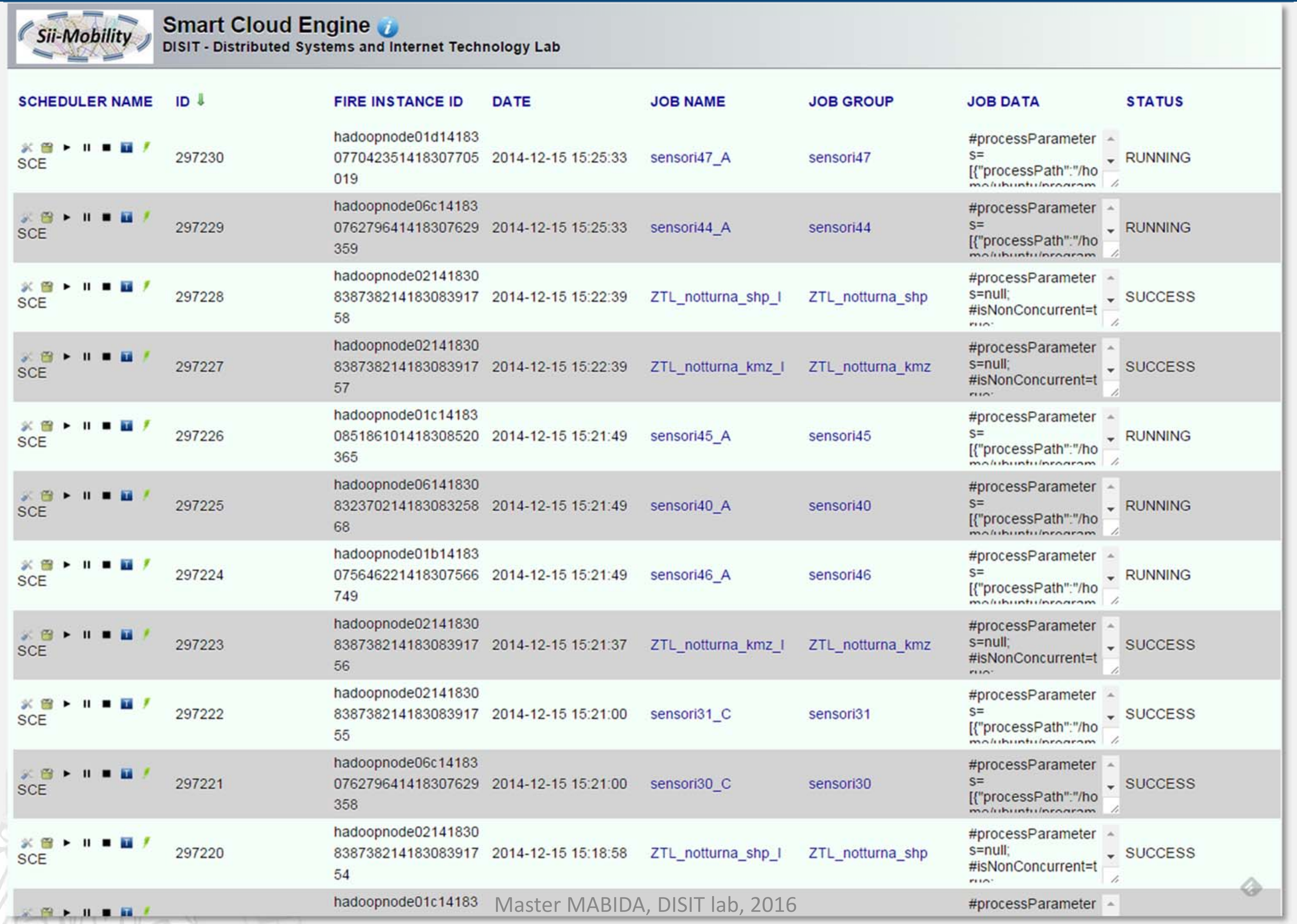




Smart Cloud Engine
DISIT - Distributed Systems and Internet Technology Lab

Id	Sla	Timestamp	Alarms	Tot
394443	urn:cloudicaro:ServiceLevelAgreement:mobmed	2014-12-16 11:54:58	4	15
394434	urn:cloudicaro:ServiceLevelAgreement:log	2014-12-16 11:53:57	1	9
394430	urn:cloudicaro:ServiceLevelAgreement:inea-lab	2014-12-16 11:52:56	2	4
394407	urn:cloudicaro:ServiceLevelAgreement:icaro-disit	2014-12-16 11:51:59	2	23
394400	urn:cloudicaro:ServiceLevelAgreement:disit-lab	2014-12-16 11:51:57	1	7
394354	urn:cloudicaro:ServiceLevelAgreement:icaro-dev	2014-12-16 11:51:01	4	46

Distributed SCE Scheduler

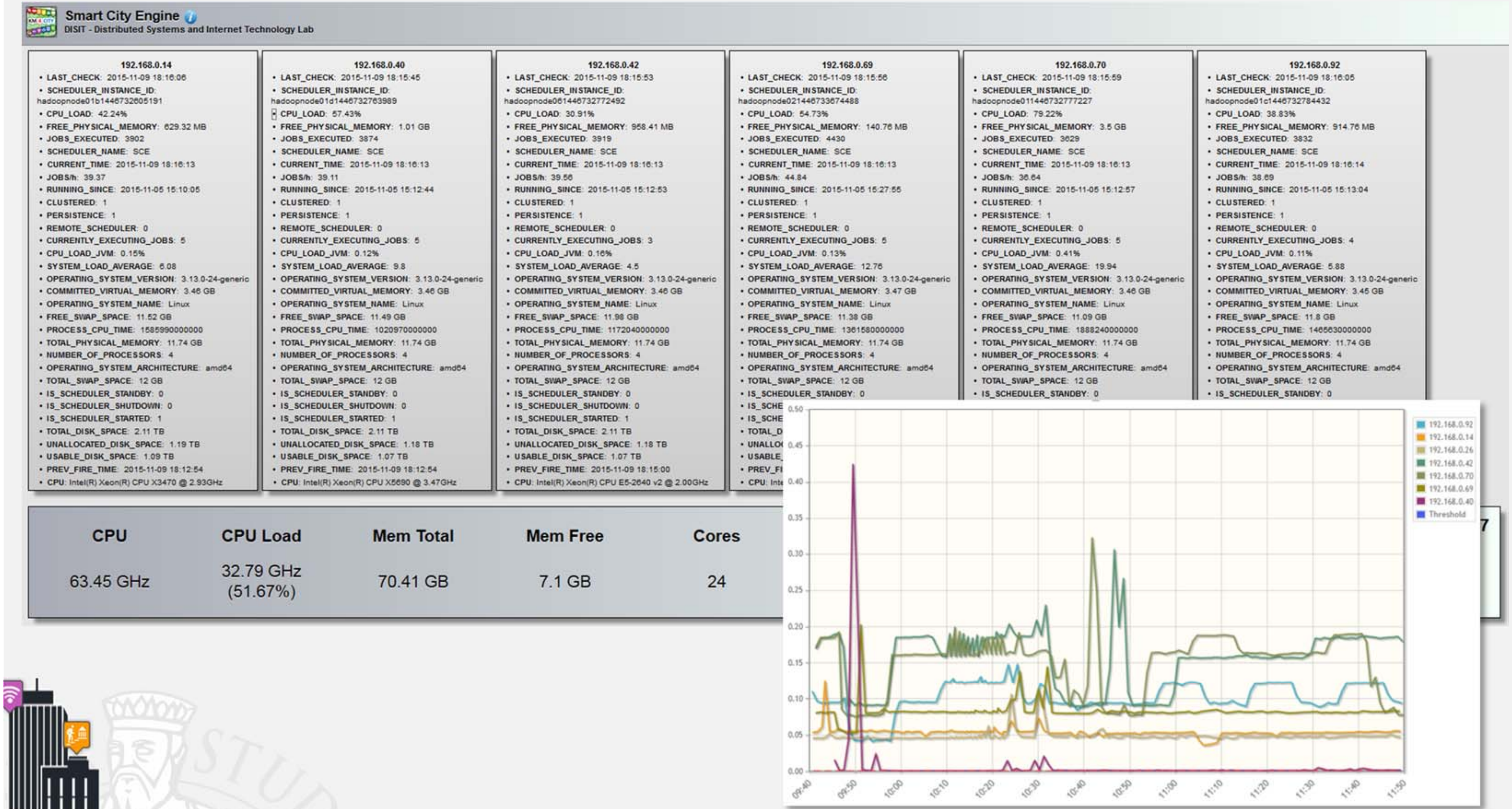


Smart Cloud Engine

DISIT - Distributed Systems and Internet Technology Lab

SCHEDULER NAME	ID	FIRE INSTANCE ID	DATE	JOB NAME	JOB GROUP	JOB DATA	STATUS
SCE	297230	hadoopnode01d14183077042351418307705019	2014-12-15 15:25:33	sensori47_A	sensori47	#processParameters= [{"processPath":"/home/ubuntu/program	RUNNING
SCE	297229	hadoopnode06c14183076279641418307629359	2014-12-15 15:25:33	sensori44_A	sensori44	#processParameters= [{"processPath":"/home/ubuntu/program	RUNNING
SCE	297228	hadoopnode0214183083873821418308391758	2014-12-15 15:22:39	ZTL_notturna_shp_I	ZTL_notturna_shp	#processParameters=null; #isNonConcurrent=true;	SUCCESS
SCE	297227	hadoopnode0214183083873821418308391757	2014-12-15 15:22:39	ZTL_notturna_kmz_I	ZTL_notturna_kmz	#processParameters=null; #isNonConcurrent=true;	SUCCESS
SCE	297226	hadoopnode01c14183085186101418308520365	2014-12-15 15:21:49	sensori45_A	sensori45	#processParameters= [{"processPath":"/home/ubuntu/program	RUNNING
SCE	297225	hadoopnode0614183083237021418308325868	2014-12-15 15:21:49	sensori40_A	sensori40	#processParameters= [{"processPath":"/home/ubuntu/program	RUNNING
SCE	297224	hadoopnode01b14183075646221418307566749	2014-12-15 15:21:49	sensori46_A	sensori46	#processParameters= [{"processPath":"/home/ubuntu/program	RUNNING
SCE	297223	hadoopnode0214183083873821418308391756	2014-12-15 15:21:37	ZTL_notturna_kmz_I	ZTL_notturna_kmz	#processParameters=null; #isNonConcurrent=true;	SUCCESS
SCE	297222	hadoopnode0214183083873821418308391755	2014-12-15 15:21:00	sensori31_C	sensori31	#processParameters= [{"processPath":"/home/ubuntu/program	SUCCESS
SCE	297221	hadoopnode06c14183076279641418307629358	2014-12-15 15:21:00	sensori30_C	sensori30	#processParameters= [{"processPath":"/home/ubuntu/program	SUCCESS
SCE	297220	hadoopnode0214183083873821418308391754	2014-12-15 15:18:58	ZTL_notturna_shp_I	ZTL_notturna_shp	#processParameters=null; #isNonConcurrent=true;	SUCCESS
		hadoopnode01c14183				#processParameter	

Distributed SCE Scheduler



Smart City Engine
DISIT - Distributed Systems and Internet Technology Lab

192.168.0.14
- LAST_CHECK: 2015-11-09 18:16:06
- SCHEDULER_INSTANCE_ID: hadoopnode01b1446732605191
- CPU_LOAD: 42.24%
- FREE_PHYSICAL_MEMORY: 629.32 MB
- JOBS_EXECUTED: 3902
- SCHEDULER_NAME: SCE
- CURRENT_TIME: 2015-11-09 18:16:13
- JOBS/h: 39.37
- RUNNING_SINCE: 2015-11-05 15:10:05
- CLUSTERED: 1
- PERSISTENCE: 1
- REMOTE_SCHEDULER: 0
- CURRENTLY_EXECUTING_JOBS: 5
- CPU_LOAD_JVM: 0.15%
- SYSTEM_LOAD_AVERAGE: 6.08
- OPERATING_SYSTEM_VERSION: 3.13.0-24-generic
- COMMITTED_VIRTUAL_MEMORY: 3.46 GB
- OPERATING_SYSTEM_NAME: Linux
- FREE_SWAP_SPACE: 11.52 GB
- PROCESS_CPU_TIME: 1585990000000
- TOTAL_PHYSICAL_MEMORY: 11.74 GB
- NUMBER_OF_PROCESSORS: 4
- OPERATING_SYSTEM_ARCHITECTURE: amd64
- TOTAL_SWAP_SPACE: 12 GB
- IS_SCHEDULER_STANDBY: 0
- IS_SCHEDULER_SHUTDOWN: 0
- IS_SCHEDULER_STARTED: 1
- TOTAL_DISK_SPACE: 2.11 TB
- UNALLOCATED_DISK_SPACE: 1.19 TB
- USABLE_DISK_SPACE: 1.09 TB
- PREV_FIRE_TIME: 2015-11-09 18:12:54
- CPU: Intel(R) Xeon(R) CPU X3470 @ 2.93GHz

192.168.0.40
- LAST_CHECK: 2015-11-09 18:15:45
- SCHEDULER_INSTANCE_ID: hadoopnode01d1446732763989
- CPU_LOAD: 57.43%
- FREE_PHYSICAL_MEMORY: 1.01 GB
- JOBS_EXECUTED: 3874
- SCHEDULER_NAME: SCE
- CURRENT_TIME: 2015-11-09 18:16:13
- JOBS/h: 39.11
- RUNNING_SINCE: 2015-11-05 15:12:44
- CLUSTERED: 1
- PERSISTENCE: 1
- REMOTE_SCHEDULER: 0
- CURRENTLY_EXECUTING_JOBS: 5
- CPU_LOAD_JVM: 0.12%
- SYSTEM_LOAD_AVERAGE: 9.8
- OPERATING_SYSTEM_VERSION: 3.13.0-24-generic
- COMMITTED_VIRTUAL_MEMORY: 3.46 GB
- OPERATING_SYSTEM_NAME: Linux
- FREE_SWAP_SPACE: 11.49 GB
- PROCESS_CPU_TIME: 1020970000000
- TOTAL_PHYSICAL_MEMORY: 11.74 GB
- NUMBER_OF_PROCESSORS: 4
- OPERATING_SYSTEM_ARCHITECTURE: amd64
- TOTAL_SWAP_SPACE: 12 GB
- IS_SCHEDULER_STANDBY: 0
- IS_SCHEDULER_SHUTDOWN: 0
- IS_SCHEDULER_STARTED: 1
- TOTAL_DISK_SPACE: 2.11 TB
- UNALLOCATED_DISK_SPACE: 1.18 TB
- USABLE_DISK_SPACE: 1.07 TB
- PREV_FIRE_TIME: 2015-11-09 18:12:54
- CPU: Intel(R) Xeon(R) CPU X5690 @ 3.47GHz

192.168.0.42
- LAST_CHECK: 2015-11-09 18:15:53
- SCHEDULER_INSTANCE_ID: hadoopnode061446732772492
- CPU_LOAD: 30.91%
- FREE_PHYSICAL_MEMORY: 958.41 MB
- JOBS_EXECUTED: 3919
- SCHEDULER_NAME: SCE
- CURRENT_TIME: 2015-11-09 18:16:13
- JOBS/h: 39.56
- RUNNING_SINCE: 2015-11-05 15:12:53
- CLUSTERED: 1
- PERSISTENCE: 1
- REMOTE_SCHEDULER: 0
- CURRENTLY_EXECUTING_JOBS: 3
- CPU_LOAD_JVM: 0.16%
- SYSTEM_LOAD_AVERAGE: 4.5
- OPERATING_SYSTEM_VERSION: 3.13.0-24-generic
- COMMITTED_VIRTUAL_MEMORY: 3.46 GB
- OPERATING_SYSTEM_NAME: Linux
- FREE_SWAP_SPACE: 11.98 GB
- PROCESS_CPU_TIME: 1172040000000
- TOTAL_PHYSICAL_MEMORY: 11.74 GB
- NUMBER_OF_PROCESSORS: 4
- OPERATING_SYSTEM_ARCHITECTURE: amd64
- TOTAL_SWAP_SPACE: 12 GB
- IS_SCHEDULER_STANDBY: 0
- IS_SCHEDULER_SHUTDOWN: 0
- IS_SCHEDULER_STARTED: 1
- TOTAL_DISK_SPACE: 2.11 TB
- UNALLOCATED_DISK_SPACE: 1.18 TB
- USABLE_DISK_SPACE: 1.07 TB
- PREV_FIRE_TIME: 2015-11-09 18:15:00
- CPU: Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz

192.168.0.69
- LAST_CHECK: 2015-11-09 18:15:56
- SCHEDULER_INSTANCE_ID: hadoopnode021446733674488
- CPU_LOAD: 54.73%
- FREE_PHYSICAL_MEMORY: 140.76 MB
- JOBS_EXECUTED: 4430
- SCHEDULER_NAME: SCE
- CURRENT_TIME: 2015-11-09 18:16:13
- JOBS/h: 44.84
- RUNNING_SINCE: 2015-11-05 15:27:55
- CLUSTERED: 1
- PERSISTENCE: 1
- REMOTE_SCHEDULER: 0
- CURRENTLY_EXECUTING_JOBS: 5
- CPU_LOAD_JVM: 0.13%
- SYSTEM_LOAD_AVERAGE: 12.76
- OPERATING_SYSTEM_VERSION: 3.13.0-24-generic
- COMMITTED_VIRTUAL_MEMORY: 3.47 GB
- OPERATING_SYSTEM_NAME: Linux
- FREE_SWAP_SPACE: 11.38 GB
- PROCESS_CPU_TIME: 1361580000000
- TOTAL_PHYSICAL_MEMORY: 11.74 GB
- NUMBER_OF_PROCESSORS: 4
- OPERATING_SYSTEM_ARCHITECTURE: amd64
- TOTAL_SWAP_SPACE: 12 GB
- IS_SCHEDULER_STANDBY: 0
- IS_SCHE
- IS_SCHE
- TOTAL_D
- UNALLOC
- USABLE_
- PREV_FI
- CPU: Inte

192.168.0.70
- LAST_CHECK: 2015-11-09 18:15:59
- SCHEDULER_INSTANCE_ID: hadoopnode011446732777227
- CPU_LOAD: 79.22%
- FREE_PHYSICAL_MEMORY: 3.5 GB
- JOBS_EXECUTED: 3629
- SCHEDULER_NAME: SCE
- CURRENT_TIME: 2015-11-09 18:16:13
- JOBS/h: 36.64
- RUNNING_SINCE: 2015-11-05 15:12:57
- CLUSTERED: 1
- PERSISTENCE: 1
- REMOTE_SCHEDULER: 0
- CURRENTLY_EXECUTING_JOBS: 5
- CPU_LOAD_JVM: 0.41%
- SYSTEM_LOAD_AVERAGE: 19.94
- OPERATING_SYSTEM_VERSION: 3.13.0-24-generic
- COMMITTED_VIRTUAL_MEMORY: 3.46 GB
- OPERATING_SYSTEM_NAME: Linux
- FREE_SWAP_SPACE: 11.09 GB
- PROCESS_CPU_TIME: 1888240000000
- TOTAL_PHYSICAL_MEMORY: 11.74 GB
- NUMBER_OF_PROCESSORS: 4
- OPERATING_SYSTEM_ARCHITECTURE: amd64
- TOTAL_SWAP_SPACE: 12 GB
- IS_SCHEDULER_STANDBY: 0

192.168.0.92
- LAST_CHECK: 2015-11-09 18:16:05
- SCHEDULER_INSTANCE_ID: hadoopnode01c1446732784432
- CPU_LOAD: 38.83%
- FREE_PHYSICAL_MEMORY: 914.76 MB
- JOBS_EXECUTED: 3832
- SCHEDULER_NAME: SCE
- CURRENT_TIME: 2015-11-09 18:16:14
- JOBS/h: 38.69
- RUNNING_SINCE: 2015-11-05 15:13:04
- CLUSTERED: 1
- PERSISTENCE: 1
- REMOTE_SCHEDULER: 0
- CURRENTLY_EXECUTING_JOBS: 4
- CPU_LOAD_JVM: 0.11%
- SYSTEM_LOAD_AVERAGE: 5.88
- OPERATING_SYSTEM_VERSION: 3.13.0-24-generic
- COMMITTED_VIRTUAL_MEMORY: 3.45 GB
- OPERATING_SYSTEM_NAME: Linux
- FREE_SWAP_SPACE: 11.8 GB
- PROCESS_CPU_TIME: 1465630000000
- TOTAL_PHYSICAL_MEMORY: 11.74 GB
- NUMBER_OF_PROCESSORS: 4
- OPERATING_SYSTEM_ARCHITECTURE: amd64
- TOTAL_SWAP_SPACE: 12 GB
- IS_SCHEDULER_STANDBY: 0

CPU	CPU Load	Mem Total	Mem Free	Cores
63.45 GHz	32.79 GHz (51.67%)	70.41 GB	7.1 GB	24

Agenda

- Big Data problems
- Big Data Pipeline of working
- Distributed and Parallel Architecture
- The grid for distributed computing
- Map Reduce Approach ←
- CAP Theorem
- Considerations

Architetture MapReduce

- At the basis of **Apache Hadoop** solution
- Designed to realize
  - Large scale distributed batch processing infrastructures
  - Exploit low costs hardware **from 1 to multiple cores, with low to large Mem, with low to large storage each**
  - Covering huge (big data) storage that could not be covered by multi core parallel architectures

- See Yahoo! Hadoop tutorial
- https://developer.yahoo.com/hadoop/tutorial/index.html

Typical problems

- Large number of nodes in the clusters
- Relevant probability of failures,
  - CPU: when hundreds of thousands of nodes are present
  - Network: congestion may lead to do not provide data results and data inputs in times
  - Network: failure of apparatus
  - Mem: run out of space
  - Storage: run out of space, failure of a node, data corruption, failure of transmission
  - Clock: lack of synchronization, locked files and records not released, atomic transactions may lose connections and consistency
- Specific parallel solutions provide support for recovering from some of these failures

Typical problems

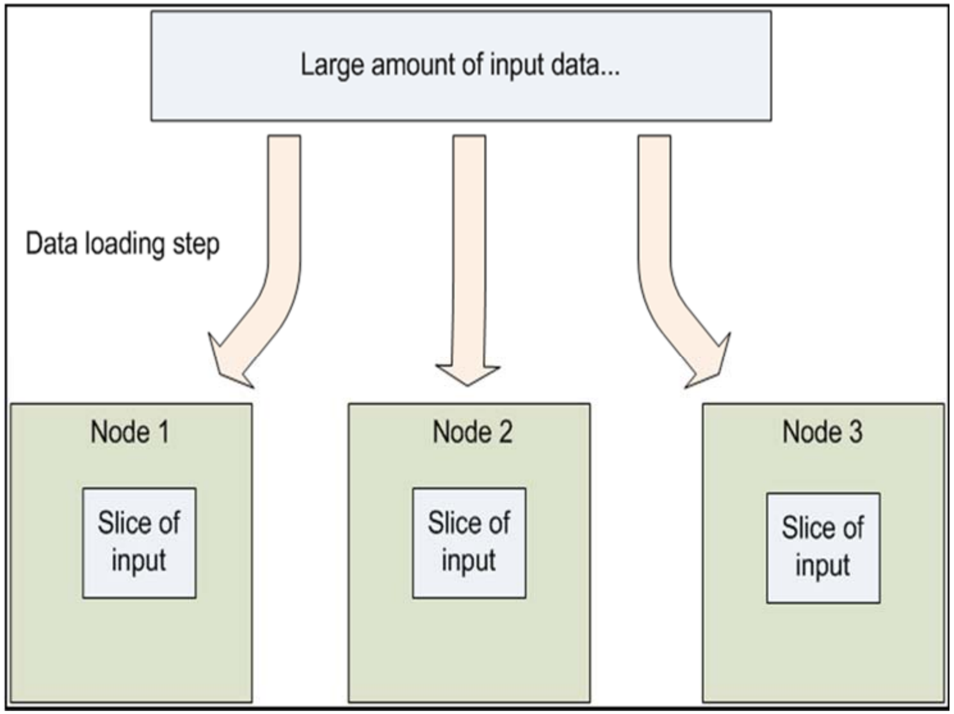
- Hadoop has no security model, nor safeguards against maliciously inserted data.
  - It cannot detect a man-in-the-middle attack between nodes
  - it is designed to handle very robustly
    - hardware failure
    - data congestion issues
- Storage may be locally become full:
  - Reroute, redistribute mechanisms are needed
  - Synchronization among different nodes is needed
  - This may cause:
    - network saturation
    - Deadlock in data exchange

Recovering from failure

- If some node fails in providing results in time, the other nodes should do the work of the missing node
- The recovering process should be performed automatically
- The Solution may be not simple to implement in non simple parallel architectures, topologies, data distribution, etc.
- Limits:
  - Individual hard drives can only sustain read speeds between 60-100 MB/second
  - assuming four independent I/O channels are available to the machine, that provides 400 MB of data every second

Hadoop data distribution

- Is based on the Hadoop Distributed File System (HDFS) that distribute data files on large chunks on the different nodes
- Each chunk is replicated across several nodes, thus supporting the failure of nodes.
  - An active process maintains the replications when failures occurs and when new data are stored.
  - Replicas are not instantly maintained aligned !!!



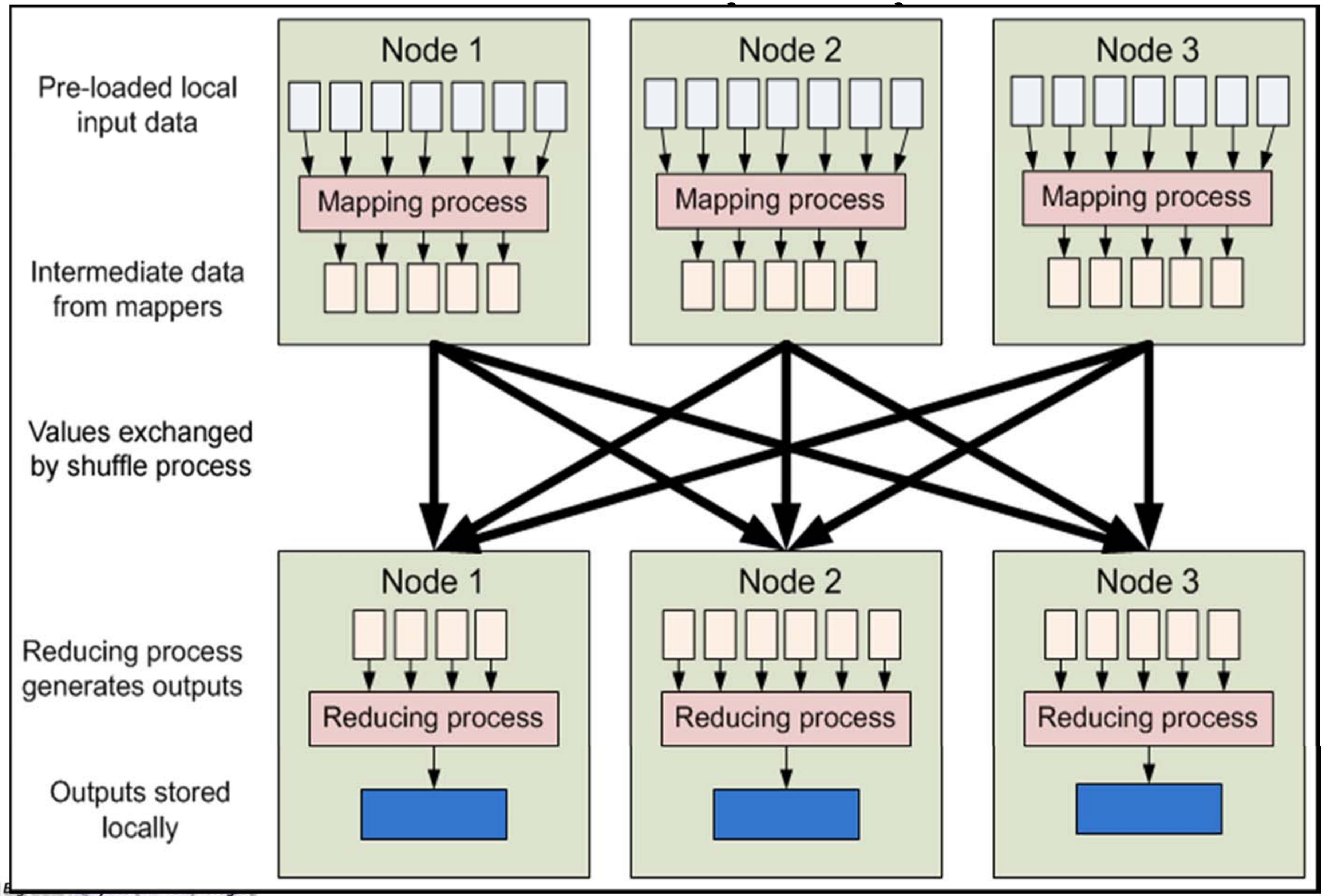
Processes vs Data

- Processes works on specific chunks of data. The allocations of processes depend on the position of the chunks they need to access,
- That is: data locality.
  - This minimize the data flow among nodes
  - Avoid communications among nodes to serve computational nodes
  - Hadoop is grounded on moving computation to the data !!! And **not** data to computation.

MapReduce: Isolated Processes

- The main idea:
  - Each individual record is processed by a task in isolation from one another. Replications allow to reduce communications for: contour conditions, data overlap, etc.

- This approach does not allow any program to be executed.
- Algorithms have to be converted into parallel implementations to be executed on cluster of nodes.
- This programming model is called MapReduce model


Node 1
Node 2
Node 3
Pre-loaded local input data
Mapping process
Mapping process
Mapping process
Intermediate data from mappers
Values exchanged by shuffle process
Node 1
Node 2
Node 3
Reducing process generates outputs
Reducing process
Reducing process
Reducing process
Outputs stored locally

Map Reduce

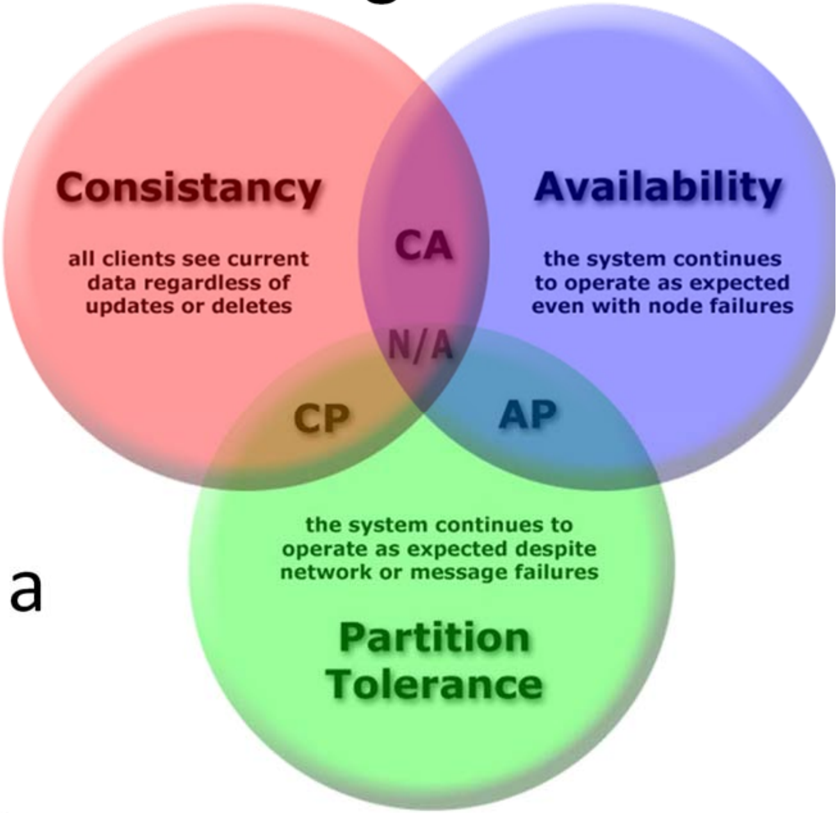
- Nelle prossime lezioni farete una lezione da 2 ore specifica su come si programma con il paradigma Map Reduce.
  - 11 Novembre
- In DISIT lab vi sono vari cluster Hadoop e vengono utilizzati per:
  - NLP massivo
  - Twitter Vigilance massivo
  - Esperimenti specifici

Agenda

- Big Data problems
- Big Data Pipeline of working
- Distributed and Parallel Architecture
- The grid for distributed computing
- Map Reduce Approach
- CAP Theorem ←
- Considerations

CAP theorem

- The **CAP theorem** (Consistency - Availability - Partition tolerance) is essential to **understand** the **behavior** of **distributed SW systems**, and **how** to **design** the **architecture** in order to meet stringent requirements, such as:
  - High **performance**.
  - Continued **availability**.
  - **Geographically distributed** systems.
- Working on billions and trillions of data every day, **scalability** became a key concept.


Consistancy
all clients see current data regardless of updates or deletes
Availability
the system continues to operate as expected even with node failures
CA
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance

CAP theorem

It is **highly desirable** for a distributed SW system **simultaneously** provide:

- **Consistency**
- Continued **Availability**
- **Partitions Tolerance**

…but this is **not possible**!!

You can satisfy **at most 2** out of this 3 **requirements**, so it is necessary to determine in each case which of these three characteristics sacrifice.

Consistency

- A distributed system is **fully consistent** if a **data** written on a **Node A** is **equal** to the value read from another **Node B** , *otherwise you get an error*.
- The system will return the last value written (consistent).

- Example - **Cache Memory**:
  - In a **single node** the total **consistency is guaranteed**, as well as the tolerance to the partitions. There is not enough availability (fault tolerance) and good performance.
  - If the cache is **distributed** on two or more nodes, the **availability increases**, but **complex mechanisms** must be provided, which allow each node to access a **virtual** distributed **repository** (to read the same value).

Availability

- A distributed system is **always available** if **each** working **node** is **always** able to **respond** to a query or provide its services, it may be not consistent with the equivalent value given from others nodes. See for example the number of likes on SN
- Example – **Cache Memory**:
  - A cache on a **single node** does **not guarantee** continuous **availability**.
  - A **distributed cache** keeps, on various nodes, some areas to store backup data of other nodes.
  - In order to realize the **continuous availability** , **data redundancy** is required (same data on multiple nodes). This requires mechanisms to ensure the consistency and to avoid problems regarding partitions tolerance.

Partitions Tolerance

- It is the ability of a system to be **tolerant to add/ remove** a **node** in a distributed system (partitioning) or to the loss of messages on the network. This may provoke inconsistencies, and in some cases fault if they are not accepted.

- Example - **clusters** formed by nodes on **two different data centers**:
  - If data center **lose** their **network connectivity**, nodes in the cluster can **no** longer **synchronize** the system state.
  - **Nodes** of the **same data center**, reorganize themselves into sub-clusters, **cutting off** node of the **other data center**.
  - The system will continue to operate in an uncoordinated manner, with **possible data loss**.

Consistency/Availability (CA)

- By designing a distributed system, you must consider as a compromise solution accept: **CA**, **CP** and **AP**.

- It is the compromise offered by **relational DBMS**.
- **Data** is **consistent** on all nodes (active and available).
- Writes/reads are always possible, updated data are propagated across the cluster nodes.
- Possible issues:
  - ☹ performance and scalability
  - ☹ misalignment between the data in the case of partitions of nodes.

Consistency/Partition-Tolerance (CP)

- **Preferred** compromise **solutions** by HBase, MongoDB, BigTable.
- **Data** is **consistent** on all nodes, **partitions** are **guaranteed**, ensuring data synchronization.

- Possible issues:
  - ☹ **Availability**: data may be no longer available if a node goes down (e.g., limited number of replicas).

Availability/Partition-Tolerance (AP)

- Compromise solutions used by **CouchDB**, **Riak**, Apache **Cassandra**.
- Nodes remain online even if unable to talk to each other.
- It **requires** a process of **data re-synchronization** to eliminate any conflicts when the partition is resolved.
- The system is still available under partitioning, but some of the **data returned** may be **inaccurate**.

☺ **Good performance** in terms of latency and scalability.

Big Data's problems

Architectures to collect, refine and analyze all this huge amount of data collected, **are inefficient**:

There is need for **innovation**!

Other critical aspects that should be consider:

- **Issues** related to the **data quality** and **reliability**.
- **Issues** related to **privacy** and **data ownership**.

Data quality and reliability

Data quality is determined by :

- **Completeness**: presence of **all** information needed to describe an object, entity or event (eg. Identifyng).
- **Consistency**: data must not be **contradictory**. For example, the total balance and movements.
- **Accuracy**: data must be **correct**, i.e. conform to actual values. For example, an email address must not only be well-formed *nome@dominio.it*, but it must also be valid and working.

Data quality and reliability

- **Absence of duplication**: tables, records, fields should be stored **only once**, avoiding the presence of copies. Duplicate information involve double handling and can lead to **problems** of **synchronization** (consistency).

- **Integrity** is a concept related to **relational databases**, where there are tools to implement integrity constraints. Example a control on the types of data (contained in a column), or on combinations of identifiers (to prevent the presence of two equal rows).

Data quality and reliability

- The overall **data quality** can be **undermined** by:
  - Errors in **data entry** operations (fields and missing information, incorrect or malformed).
  - Errors in **data management** software (query and incorrect procedures).
  - Errors in the **design** of databases (conceptual and logical errors).

Data quality and reliability

- In the world of Big Data instead:
  - **Operational data**: the quality problems are known and there are several tools for **automatic** data **cleansing → be careful of losing information**.
  - **Data automatically generated**: scientific data and data from sensors, have **no entry errors**, but they are "**weak**" in terms of information content: there is the need to **integrate data** from other systems and then **analyze** them.
  - **Data on the Web**: Social networks, forums, blogs generate **semi-structured data**. The most reliable are the **metadata** (if present) shall instead be subject to errors, abbreviations, etc.

Data quality and reliability

- **Disambiguate information**: the **same data** can have **different meanings**. The challenge is trying to find the most relevant to the present context. Help are tags, tagging the data you are trying to highlight the scope of relevance.

- **Truth**: News, statements, documents do **not always** correspond to **reality** or real.

- The quality of the data, however, is also **linked** to the **context** in which they are analyzed. Transactions of filtering and **cleaning** must be done **by degrees** to **avoid removing** potentially useful **data**.

Privacy and data ownership

- The Big Data problems are connected to **privacy**, ownership and **use** of **data** by third parties.
- **Data of the Web**: the user-generated-content are shared for all. It is **ethical** usage?
- **Sensitive data**: the data in the DB hospitals regarding the **medical history** of the patients, are properly protected?
- **Location Data**: use of smartphones, GPS, electronic payment systems, but also social networks **leave traces** of where you can get the movements of the user.

Agenda

- Big Data problems
- Big Data Pipeline of working
- Distributed and Parallel Architecture
- The grid for distributed computing
- Map Reduce Approach
- CAP Theorem
- Considerations ←

Considerazione 1

- La maggior parte degli **algoritmi** che avete visto a statistica o di machine learning:
  - Non **sfruttano l'architettura parallela** sottesa… se mai ne do una
  - Vanno **totalmente rivisti in chiave di architettura parallela** quando si passa alla loro esecuzione su architettura big data.
- L'alternativa è elaborare una mole molto minore dei dati (scelti in qualche modo) e supporre che tutto poi si applichi anche ai big data ?
  - Questo non è detto che vada bene per tutti i problemi:
    - Per esempio non va bene per il detection di casi critici, che sono tipicamente casi sporadici e rischiano di essere persi nel filtering o nel quality improvement, o nella ripulitura
    - Non va bene per raggiungere precisioni in predizione elevate

Considerazione 2 a

- La maggior parte degli **Strumenti** che avete visto fino adesso (e.g., Excel, database SQL, R):
  - Non lavorano su architetture parallele → NON sono in grado di sfruttare un sistema distribuito e/o parallelo di calcolatori
- L'alternativa è elaborare una mole molto minore dei dati (scelti in qualche modo) e supporre che tutto poi si applichi anche ai big data ?
- Questo non è detto che vada bene per tutti i problemi:
  - non va bene per i database se i dati non ci stanno non ci si mettono e siamo bloccati, il resto che si fa si **BUTTA** ?
  - Non va bene per i modelli di apprendimento o che altro.. Si rischia prendendo meno dati di risolvere un **problema DIVERSO** da quello originale e non è detto che i due problemi abbiano la stessa soluzione.

Considerazione 2 b

- Vanno utilizzati **strumenti diversi o evoluti**
  - R usa una sola CPU, provate MRO, Microsoft R e non tutte le funzioni sono presenti.
  - R per Hadoop non è semplice da installare e non tutte le funzioni sono presenti, molte vanno rifatte in termini di Map Reduce.
  - Tipicamente vanno utilizzate soluzioni ad-hoc, solo le soluzioni ad-hoc funzionano con i bigdata, in molti casi, i ***tool commerciali e/o visuali servono solo per fare palestra***.

Come si Opera

- Si analizza il problema in un contesto ristretto (small data vs big data)
  - 1) Data transformation
  - 2) PCA, statistica descrittiva, etc..
- Si sviluppa un modello (predittivo, di analisi, etc. in un contesto ristretto, etc..)
  - Si opera una validazione, una valutazione del modello
- Si valuta i requisiti di computazione del problema, con che tempi, chi lo deve usare, che tipo di risultato deve accedere, etc.
- Si scala in architettura parallela usando big data operando dei compromessi su algoritmi e data processing.
- Si sviluppa Data Visualization Tool specifici

Architetture Cloud

- La maggior parte delle soluzioni Big Data sono su cloud
- Come funziona il cloud, quali sono i concetti primari?
  - – Ne parliamo l’11 Novembre

NoSQL database

- I database tradizionali SQL non sono adatti per gestire grosse moli di dati:
  - Problematiche degli SQL
  - NoSQL database
    - RDF graph database oppure Tabular etc.
  - Modelli federati
- Di questo ne viene parlato nelle prossime lezioni
  - 18 Novembre 2016