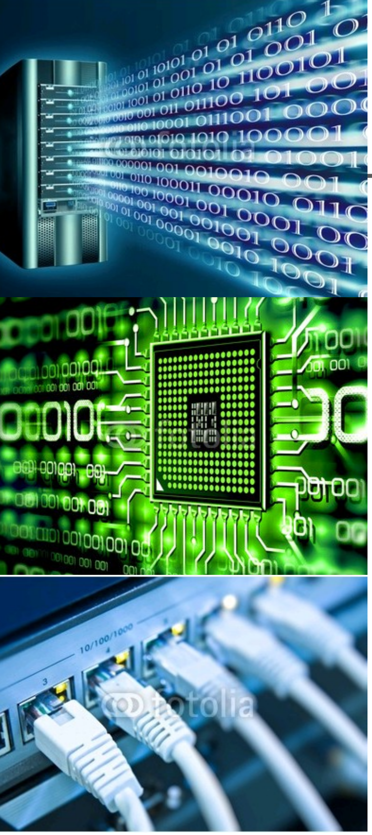


hurence

Hybrid Software Architectures for Big Data

Laurence.Hubert@hurence.com
@hurence
http://www.hurence.com

Headquarters : Grenoble

Pure player

Training

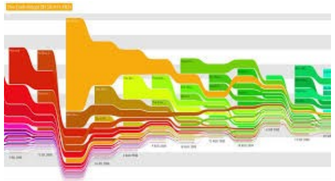
"hot-line" assistance

Expert level consulting

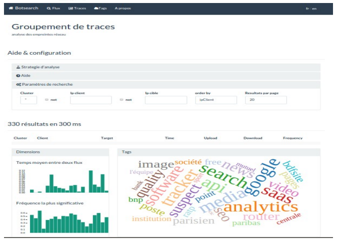
R&D Big Data X-data

XDATA
START UPS
DATA PUBLICA
cinequant
hurence
GRANDS GROUPES
orange
LE GROUPE LA POSTE
VEOLIA
esri
EDF
INSTITUTS DE RECHERCHE
TELECOM ParisTech
Inria

Products



B-DAP
Big Data Analytics
Platform



Groupement de traces
analyse des empreintes réseau

Aide & configuration

Strategie d'analyse
Aide
Paramètres de recherche

Cluster		Ip client		Ip cible	order by	Resultats par page
*	□ not		□ not		ipClient	20

330 résultats en 300 ms

Cluster	Client	Target	Time	Upload	Download	Frequency

Dimensions
Temps moyen entre deux flux
Fréquence la plus significative

Tags

BotSearch
Bot detection in
Non supervised
way

Big Data versus High Performance Computing

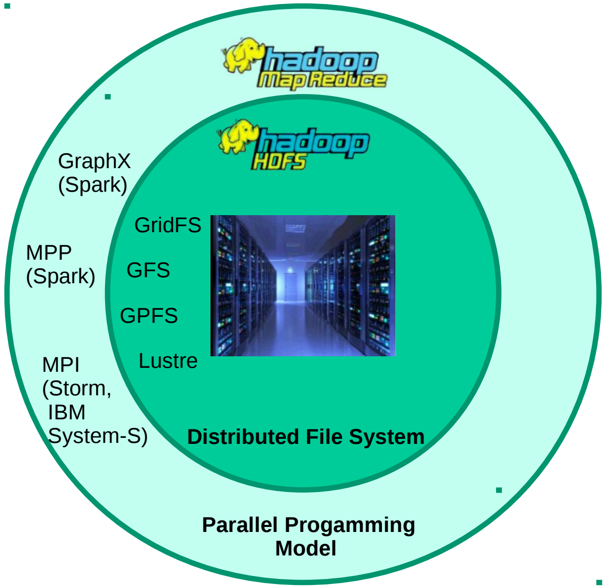
- Big Data inherits all the concepts and architectures designed by HPC experts
    - High processors density
    - IO optimizations (avoid network latency by co-locating tasks and data)
    - High availability (fail over mechanisms) to ensure no down time
    - Etc.
- Big Data democratizes HPC and make it enter traditional Information Systems
    - Still generally less advanced than specialized HPC systems with GPUs and FPGAs etc.
    - More heterogeneous from a software standpoint: various tools coming from different open source communities and not initially designed to work together.
    - Not so much focused on power consumption at the moment...
    - More focused on providing easy enough installation and monitoring tools as well as programming tools and high level environments for **non programmers (the Business Intelligence or Marketing teams).**

Big Data benefits from HPC computing’s revisited software…

Scale-out approach with commodity hardware (versus scale-up)

A distributed File System

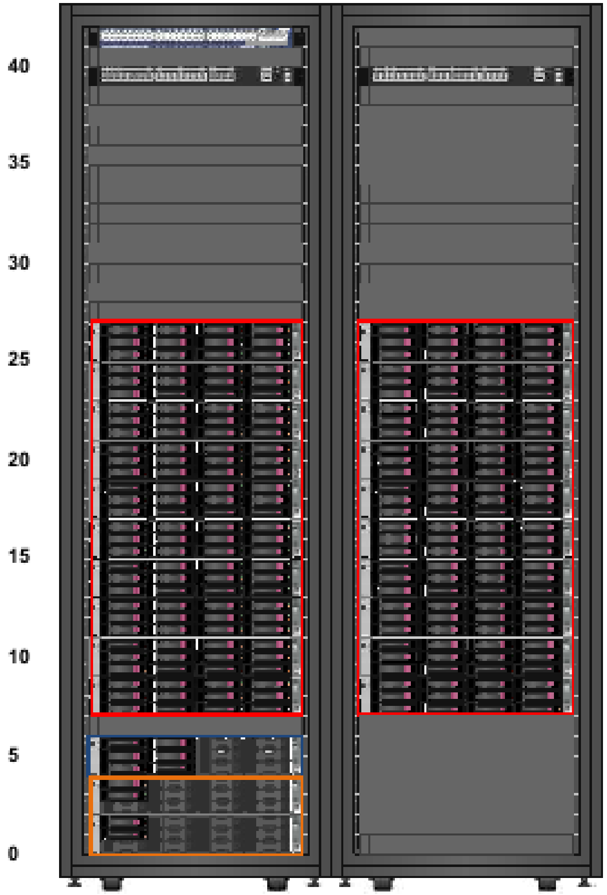
- Store data on several machines (**High availability**)
- Replicate data several times (**fault tolerance**)



A parallel programming model

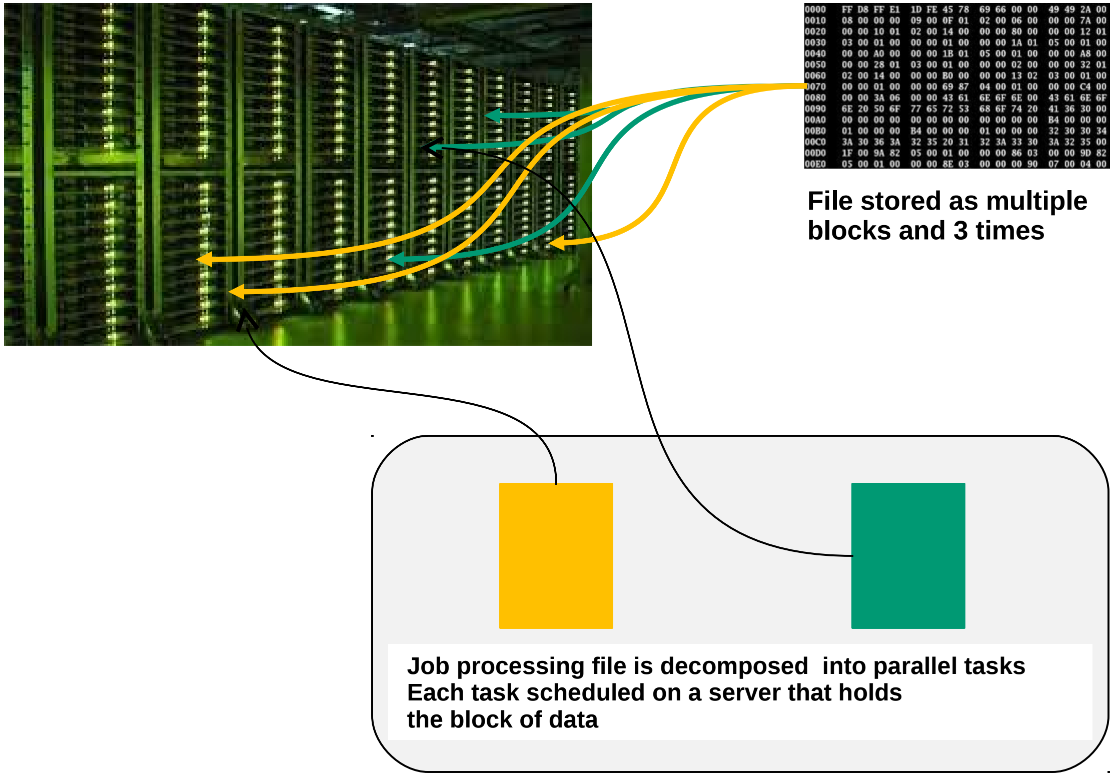
- Organize tasks on multiple machines (**parallelization**)
- Schedule tasks where the data is (**no network latency**)

Big Data star: the Hadoop cluster



- 2 racks (on different electric systems)
- Network switch of 10 gigabytes / second to connect machines
- A number of machines to act as “data nodes” (store data) and “task nodes” (compute things).
  - 2 processors machines (2x4 or 2x6 or 2x8 cores)
  - Not less than 6 gigabytes RAM/core
  - DAS Storage (Directly Attached Storage) with 1-2-3-4 terabytes disks SAS or SATA configured as JBOD: Just a Bunch of Disk

Big Data / Hadoop Distributed File System principles



```
0000    FF D8 FF E1  1D FE 45 78  69 66 00 00  49 49 2A 00
0010    08 00 00 00  09 00 0F 01  02 00 06 00  00 00 7A 00
0020    00 00 10 01  02 00 14 00  00 00 80 00  00 00 12 01
0030    03 00 01 00  00 00 01 00  00 00 1A 01  05 00 01 00
0040    00 00 A0 00  00 00 1B 01  05 00 01 00  00 00 A8 00
0050    00 00 28 01  03 00 01 00  00 00 02 00  00 00 32 01
0060    02 00 14 00  00 00 B0 00  00 00 13 02  03 00 01 00
0070    00 00 01 00  00 00 69 87  04 00 01 00  00 00 C4 00
0080    00 00 3A 06  00 00 43 61  6E 6F 6E 00  43 61 6E 6F
0090    6E 20 50 6F  77 65 72 53  68 6F 74 20  41 36 30 00
00A0    00 00 00 00  00 00 00 00  00 00 00 00  B4 00 00 00
00B0    01 00 00 00  B4 00 00 00  01 00 00 00  32 30 30 34
00C0    3A 30 36 3A  32 35 20 31  32 3A 33 30  3A 32 35 00
00D0    1F 00 9A 82  05 00 01 00  00 00 86 03  00 00 9D 82
00E0    05 00 01 00  00 00 8E 03  00 00 00 90  07 00 04 00
```

File stored as multiple blocks and 3 times

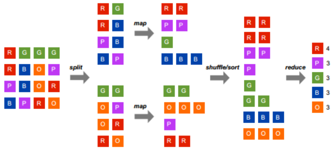
Job processing file is decomposed into parallel tasks
Each task scheduled on a server that holds
the block of data

REPLICATION

CO-LOCATION OF TASKS AND DATA

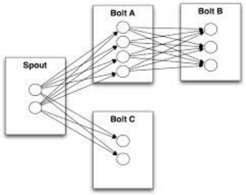
Parallel programming models

Batch or « near real time

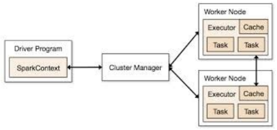


- **Map Reduce**

Real time



- **MPI** (Message Passing Interface – stream processing)
- **MPP** (Massively Parallel Processing)



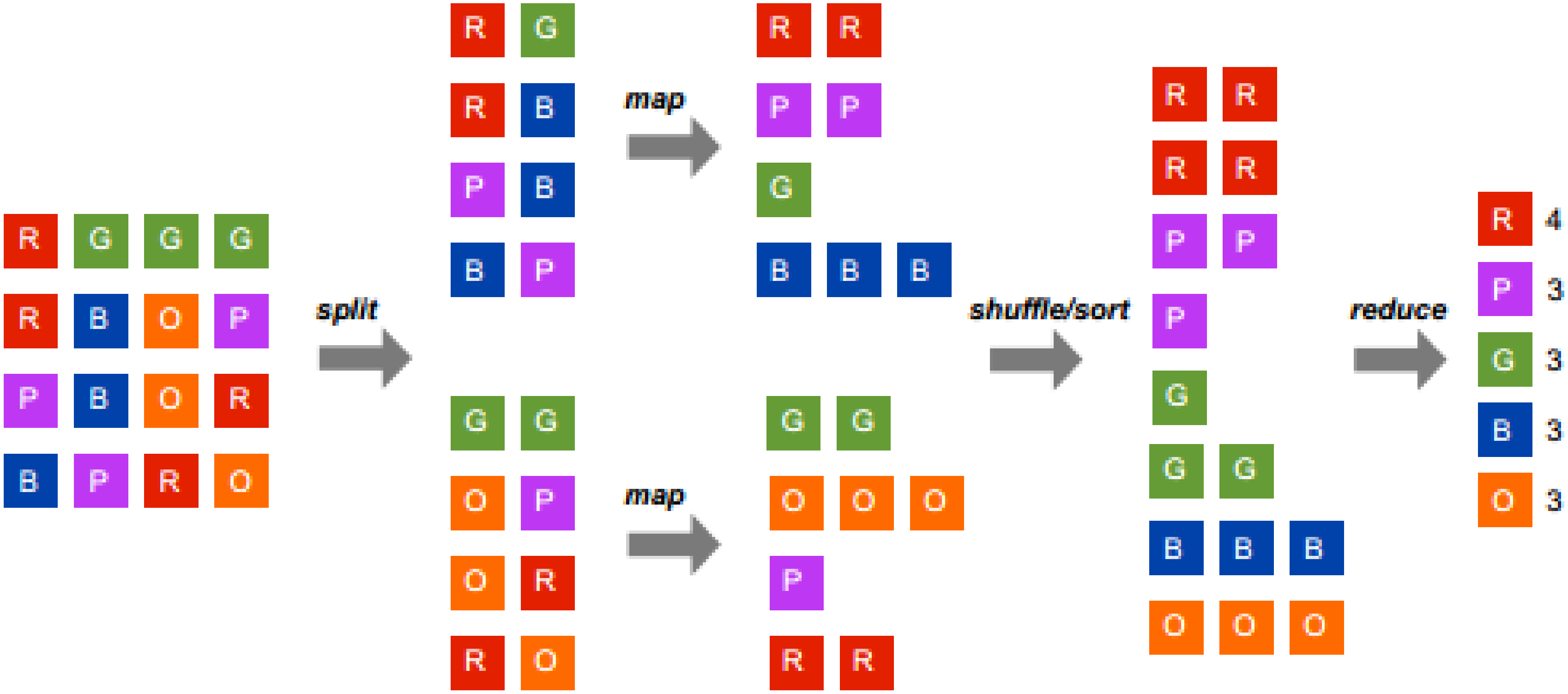
Specialized

- **GraphX** to process graphs on top of MPP

Since Hadoop YARN 2.0 Hadoop supports all models and is becoming a de-facto **Big Data operating system**

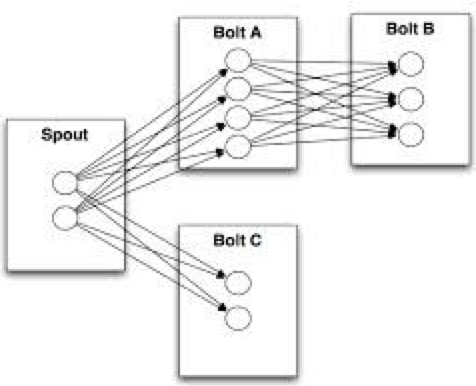
Understanding Map Reduce through Hadoop MR

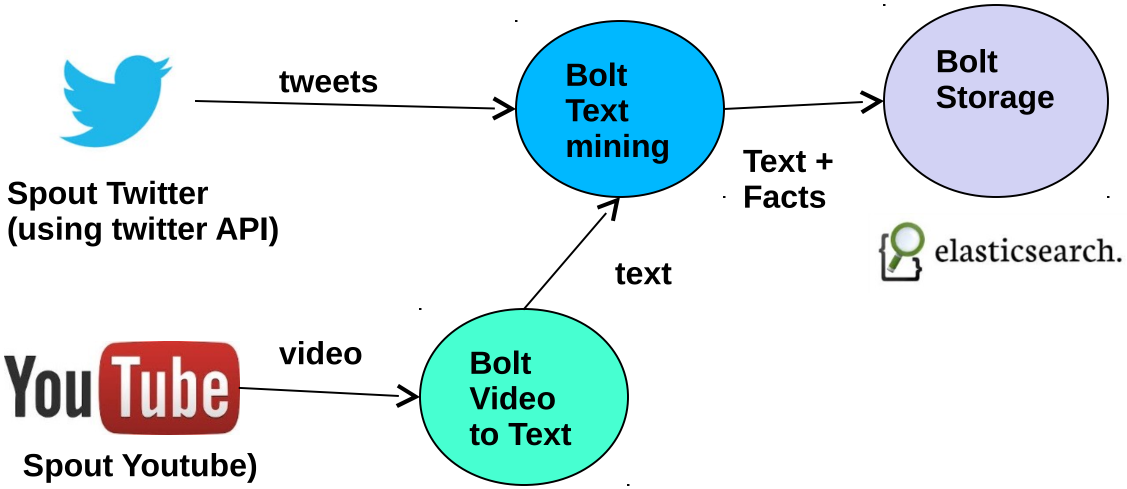
Counting colored squares…



Understanding MPI (Message Passing Interface) through Storm

Jobs are DAGs
(Directed Acyclic Graphs)
of tasks with 2 types:
Spouts: sources of data
Bolts: processors of data





tweets

Bolt Text mining

Bolt Storage

Spout Twitter
(using twitter API)

Text + Facts

elasticsearch.

text

video

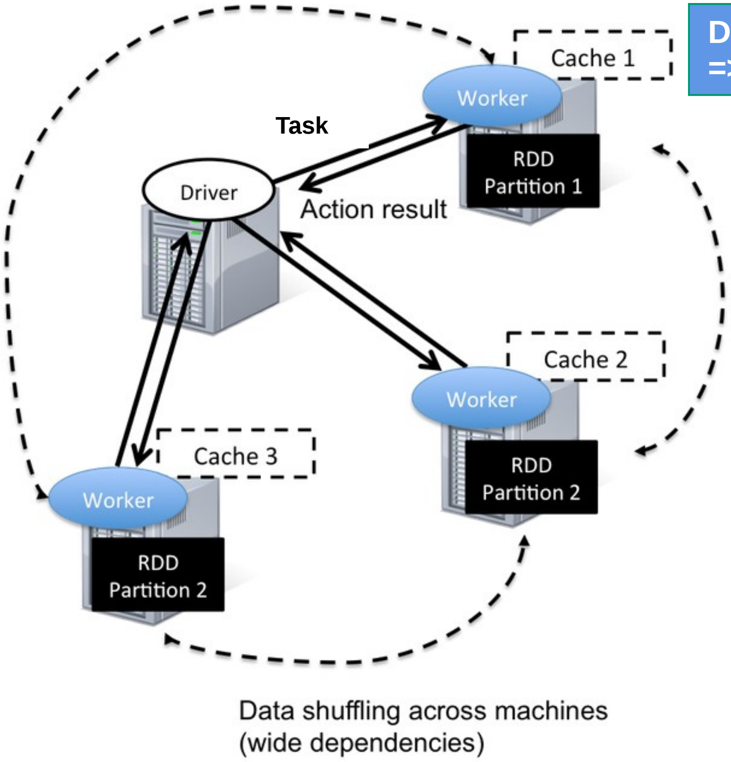
Bolt Video to Text

Spout Youtube)

Understanding MPP (Massively Parallel Processing) with Spark

Low latency, real-time, in-memory parallel processing…

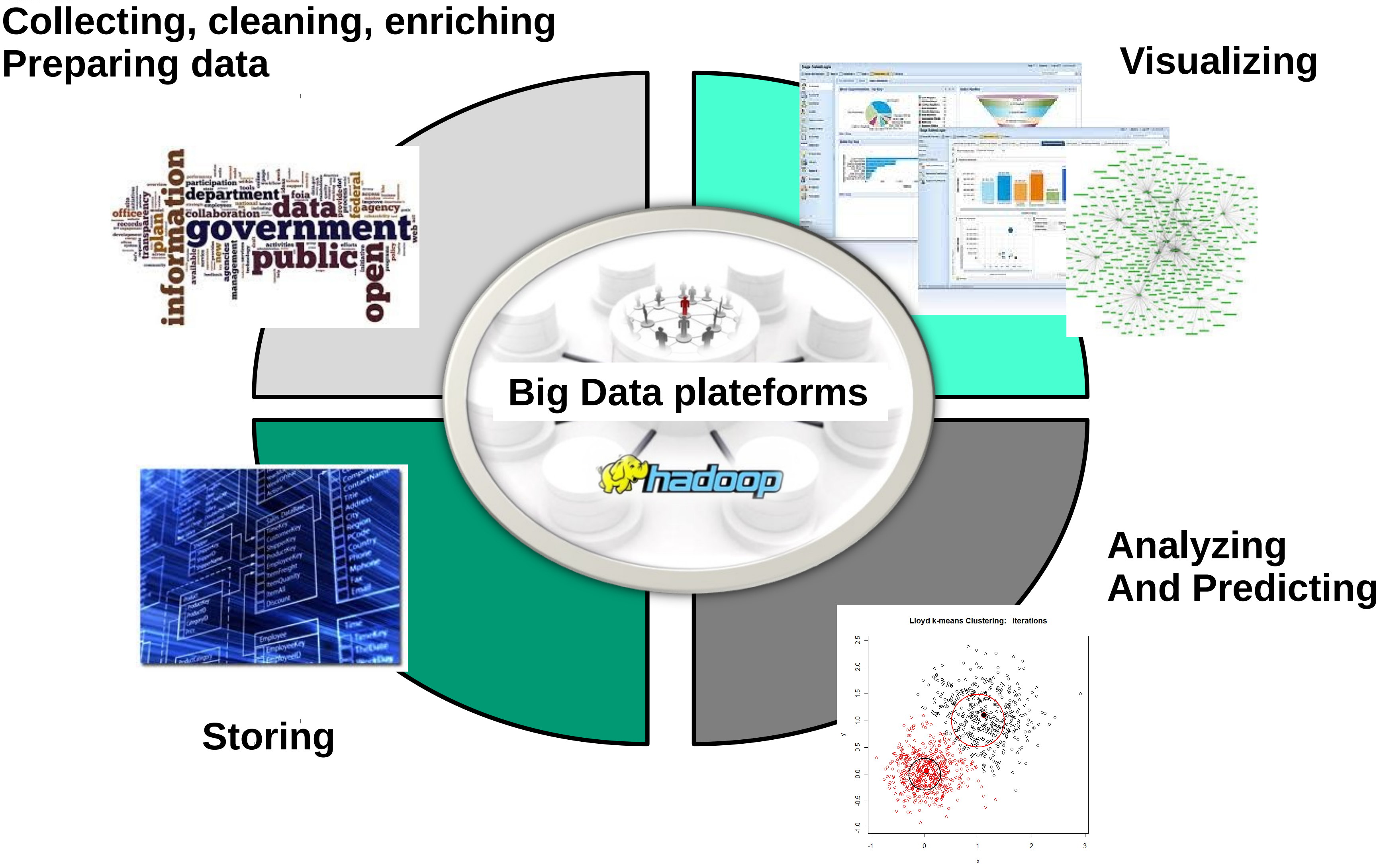
A program is decomposed into parallel tasks positioned on machines
First iteration, no data is cached
Second iteration, the program may run on cached data sets.



Data is cached in-memory…
=> Distributed cache

RDD: Resilient Distributed Dataset

Big Data Life Cycle and Big Bazaar!



Collecting, cleaning, enriching
Preparing data

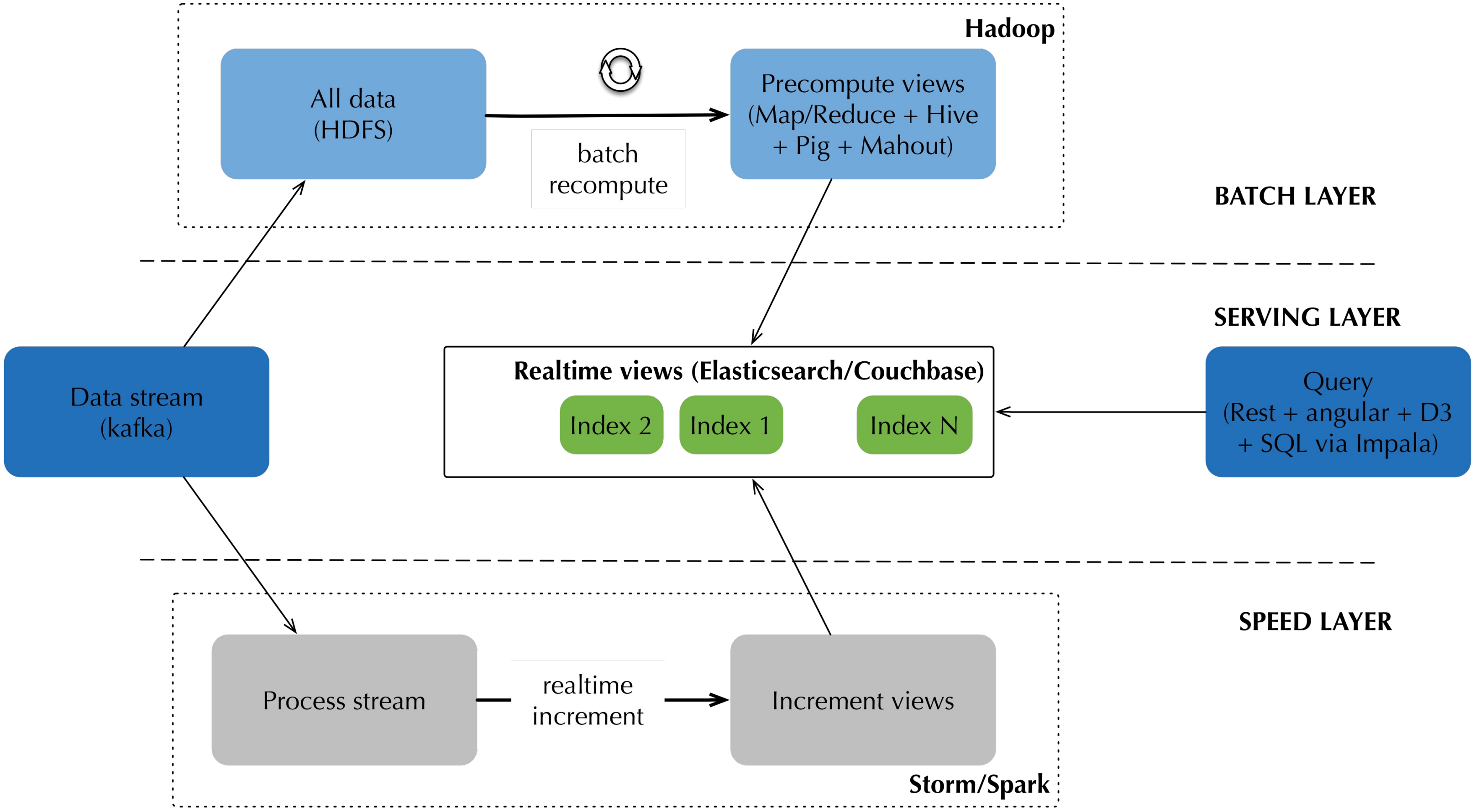
Visualizing

Big Data plateforms

Storing

Analyzing
And Predicting

ƛ (lambda) architectures



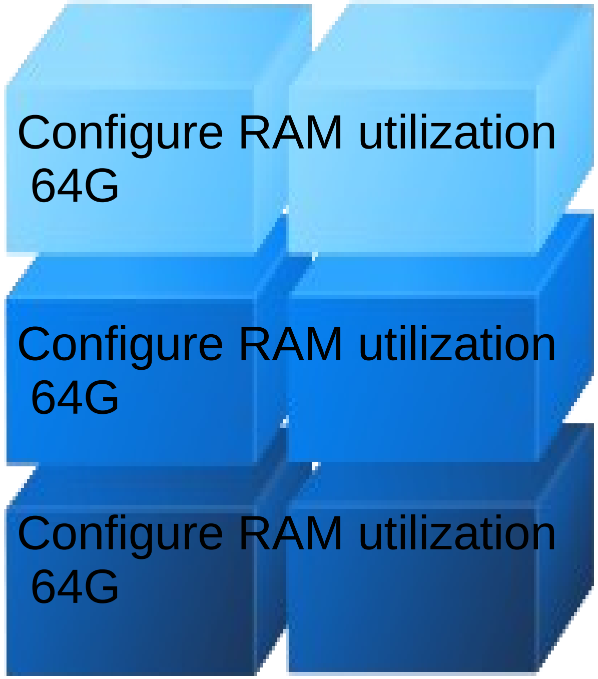
New architectures enabled by Hadoop YARN 2.0

Different job schedulers can now launch different jobs (batch jobs or real-time jobs)

Resources are managed globally by a resource manager

Still... some layers / tools do not release their resources if not needed ; they are not good multi-tenant citizen...

Different parallel paradigms want to access the same resources



Configure RAM utilization 64G

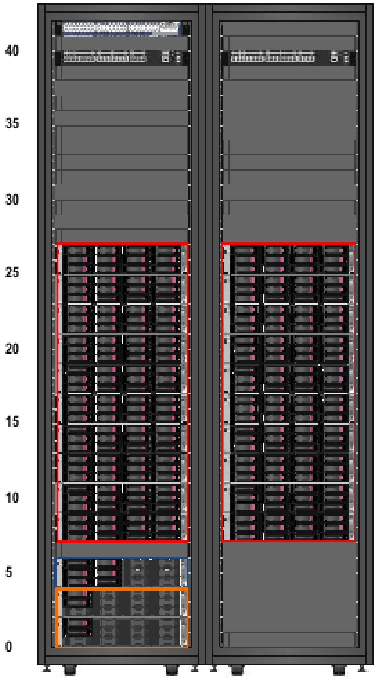
Configure RAM utilization 64G

Configure RAM utilization 64G

I am the Speed layer
and I want all my RDD
to be in-memory

I am the service layer and
I want to cache the database
data in-memory

I am the batch layer and I
want to set maximum memory
on my processes to avoid
swapping since swapping
is performance killer



For network latencies reasons all goes to the same cluster of machines

The problem with superposing many Big Data technologies

We have an **inflation of memory** on the machines… because we must provide enough memory for each of the components including databases **in silos**…

An example: two different in-memory systems using the same file will load the data in-memory two times… there is no global knowledge across tools that this data is already in-memory...

Also if one layer is unoccupied, the other layers cannot use the memory it does not use in a flexible and dynamic way (there is no “global capacity scheduler”). Every single tool has a static memory configuration.

=> heterogeneity of Big Data requires better management of resources (in particular memory resources)

The way forward...

=> there is a need for very clever resources managers and schedulers sufficiently “standardized” to allow many technologies to be working together and not in silos from a hardware standpoint.

=> Hadoop YARN has been a first move towards providing common resource management ; but many improvements are needed to manage resources in a much more clever way.

Rendez-vous in 2015 for the Hadoop improvements...