Introduction to BigData – Spark – Processing

Concrete Examples Experience at SG

Arnaud Nauwynck

Oct 2022

Objectives

4 x 2 hours of lessons
4 x 2 hours of Hands-On

Objective Lesson 1 : Overview

What is BigData ?
What is Spark ?
What is Data processing ?
… concrete examples from experience in a company

Objective Lesson 2,3,4 :   Deep Dive in Spark

Outline

- What is BigData ?
  - Order of Magnitudes for « Big »
  - History
  - Evolution of Softwares, Spark
- Description of a Datalake
  - Content, data feeding
  - How it is organized
  - Who uses it
- Example of Data-Processing
  - RAW to LAKE, Reports
- Change Storage-Compute, Evolution to Cloud

About Me

Arnaud Nauwynck

2 Tweets

NOT using Twitter/LinkedIn/FaceBook/

Only arnaud.nauwynck@gmail.com

IT Passionate since 20 years

Working at Société Générale (La Defense)
In BigData team since 4 years

Expert Java + BigData + more

Github repos

https://github.com/Arnaud-Nauwynck

http://arnaud-nauwynck.github.io/

WARNING

ANY PROPS USED IN THIS VIDEO THAT SHOW RESEMBLANCE TO ANY ILLEGAL MATERIALS ARE MERELY PROPS AND SHOULD NOT BE TAKEN SERIOUSLY. DONT TRY THIS AT HOME.

DISCLAIMER

I do not know everything. I am biaised with what I know.
My opinions do not represent any company

About You …
Quick Survey ?

Raise your Hands…

You are developper ?

In Java ? SQL ?

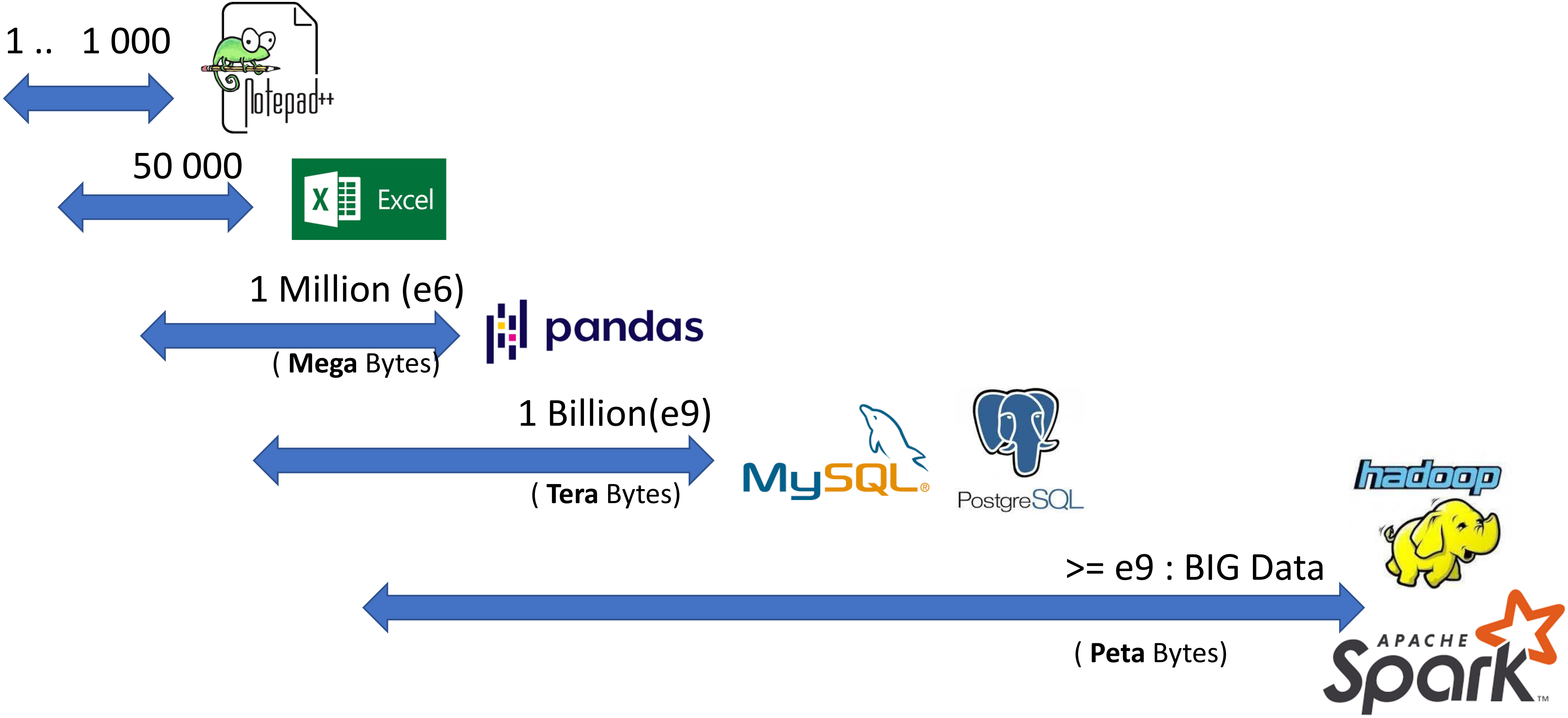
You know Hadoop / Hdfs / Spark ?

You want to be Data Engineer / Data Scientist ?

Outline

- What is BigData ?
  - Order of Magnitudes for « Big »
  - History
- Description of a Datalake
  - Content, data feeding
  - How it is organized
  - Who uses it
- Example of Data-Processing
  - RAW to LAKE, Reports
- Change Storage-Compute, Evolution to Cloud

How Big ? => Then Which Tool



1 ..   1 000

50 000

1 Million (e6)

(**Mega** Bytes)

1 Billion(e9)

(**Tera** Bytes)

>= e9 : BIG Data

(**Peta** Bytes)

Compute Resources
Vertical Scaling

Historically:

have Bigger needs => buy BIGGER Server

Very Frequent in Bank OLTP Databases

128 cores x 500Go RAM + SAN Bay of Disks

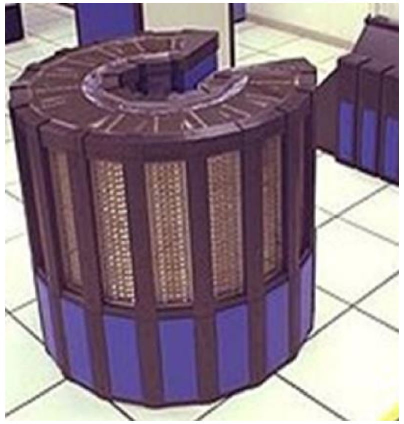
= Millions of Euros
.. BUT price is exponential to performance, and limited

BigData definition

Data does not fit on 1 server … even Huge, because of

3 V = Volume / Velocity / Variety

Bigger Hardware, different Software
Vertical Scaling -> Horyzontal



1 OS, 1 process, N Threads

Distributed computing
… Sotfwares need
Network, Fault Tolerant

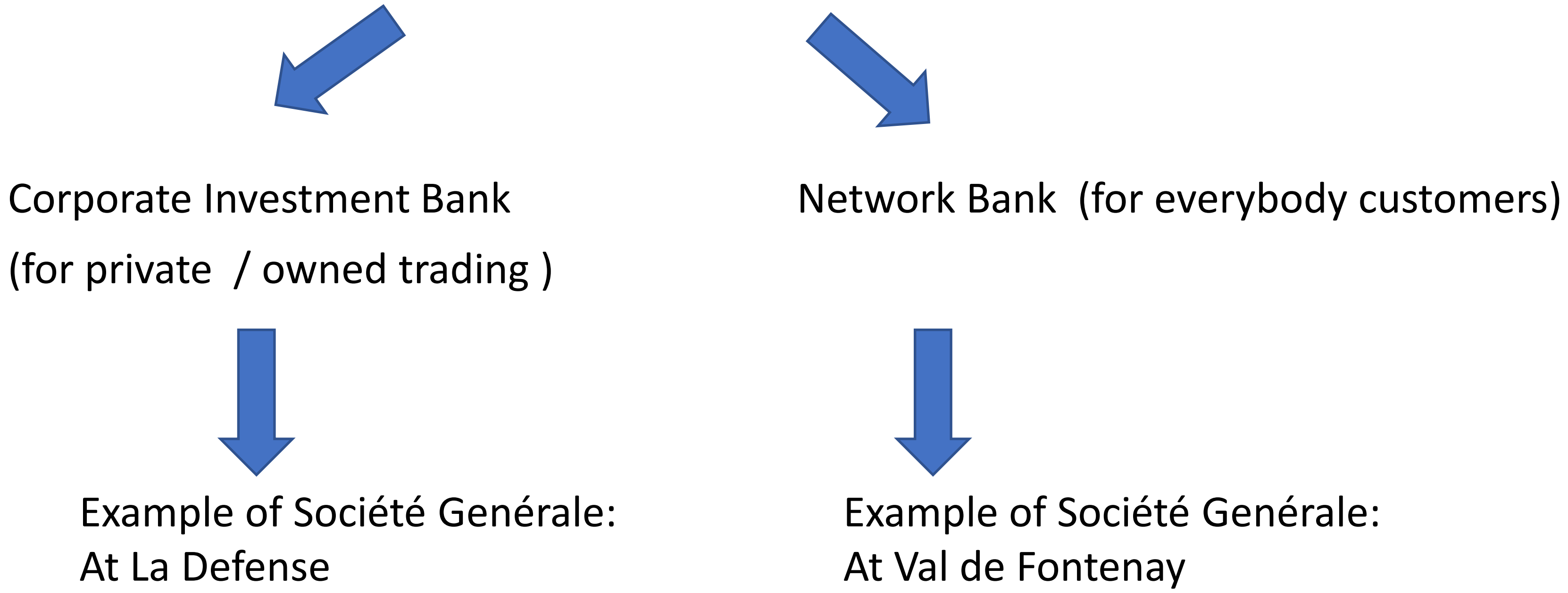
BigData At Scale
Google Datacenter

Not Only Gafas

Example of Smaller Companies, obviously doing BigData

Criteo (internet advertisment)
Cern (particule accelerators)
Gouvernment
Social Networks
Auchan, Carrefour, .. (E-Commerce)
MeteoFrance (Computing, Satelite Images)
Uber, Taxis, Cars (IOT)
Medical, Pharmaceutic, Genetics
Industries
Banks
..

BigData In Banks ?



Corporate Investment Bank

(for private / owned trading)

Example of Société Genérale:
At La Defense

Network Bank (for everybody customers)

Example of Société Genérale:
At Val de Fontenay

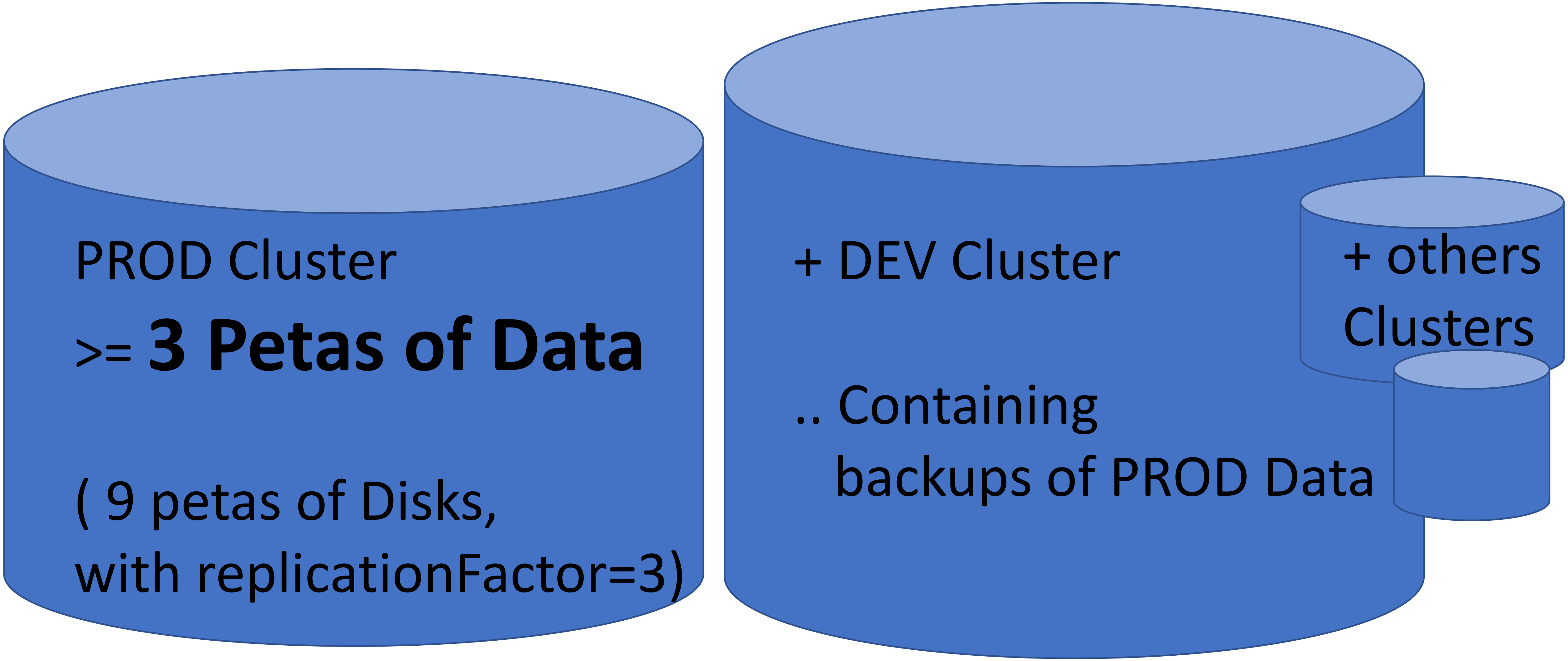
Data for

- Historical Market datas
- Risk / Var management
- Aggregation of all Trading departments
- Regulatory Reporting
- Business analysis
- ..

Data for

- RGPD… warn personnal data!
- Anti Fraud detection,
- customers-oriented
- Business analysis
- ..

BigData at SG (Investment Bank)



PROD Cluster

>= **3 Petas of Data**

(9 petas of Disks,
with replicationFactor=3)

+ DEV Cluster

.. Containing
backups of PROD Data

+ others
Clusters

Typical Datacenter Hardware
= N Racks x 10 blades (1U / 2U)

Example of a Server for Datalake

1 server = 42 cores + 256 Go RAM + 8 disks + network 1Gb/s

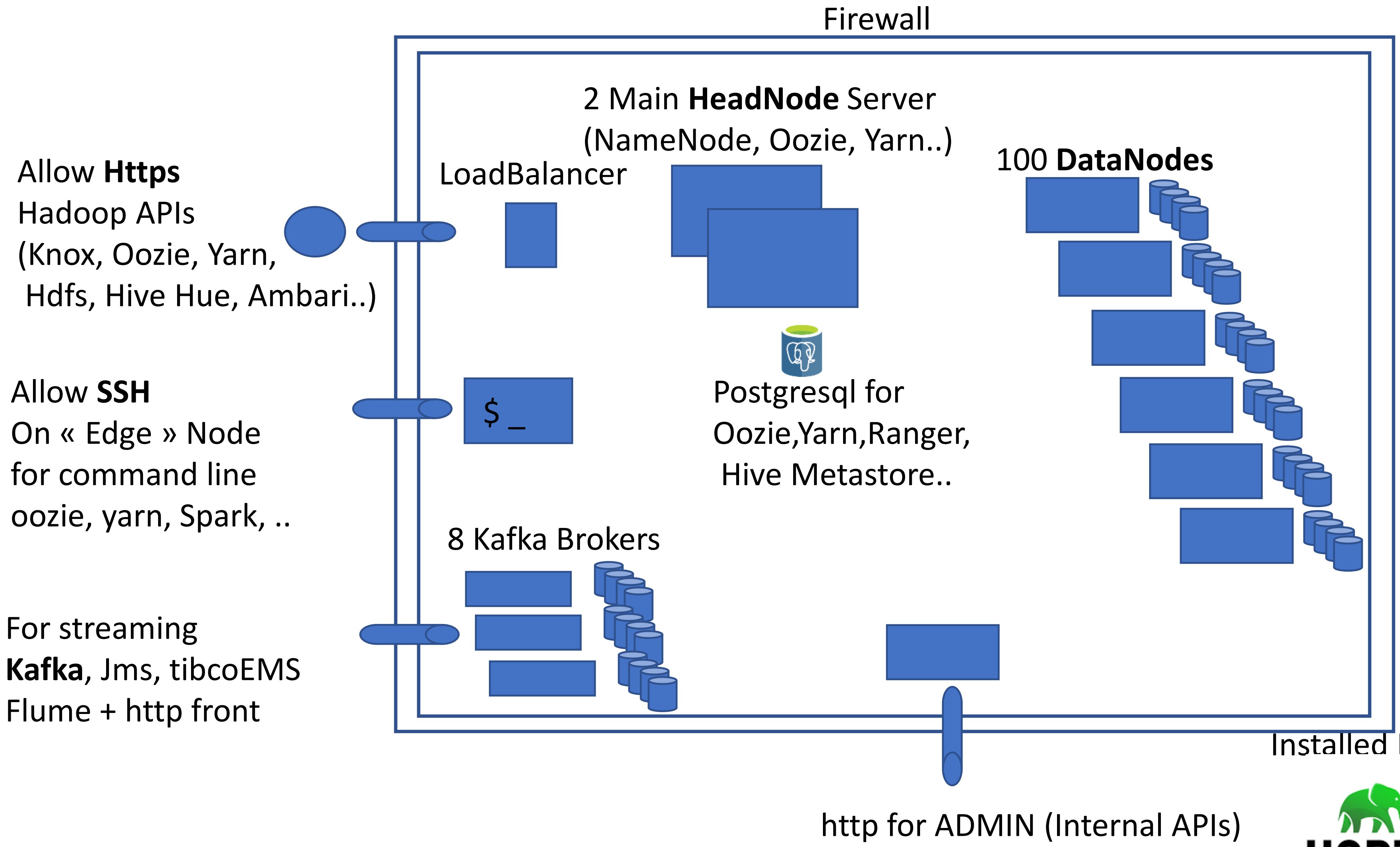
« Small » Datalake ?

100 servers for PROD
(+ 120 servers for DEV)

= 4200 cores
+ 25600 Go RAM : 25 Tera of RAMS
+ 800 disks : 2 Petas of Data (6 peta of Disks)

… now 3 Petas on Azure

Hadoop Cluster Architecture



Firewall

Allow **Https**
Hadoop APIs
(Knox, Oozie, Yarn,
Hdfs, Hive Hue, Ambari..)

LoadBalancer

2 Main **HeadNode** Server
(NameNode, Oozie, Yarn..)

100 **DataNodes**

Allow **SSH**
On « Edge » Node
for command line
oozie, yarn, Spark, ..

$ _

Postgresql for
Oozie,Yarn,Ranger,
Hive Metastore..

8 Kafka Brokers

For streaming
Kafka, Jms, tibcoEMS
Flume + http front

http for ADMIN (Internal APIs)

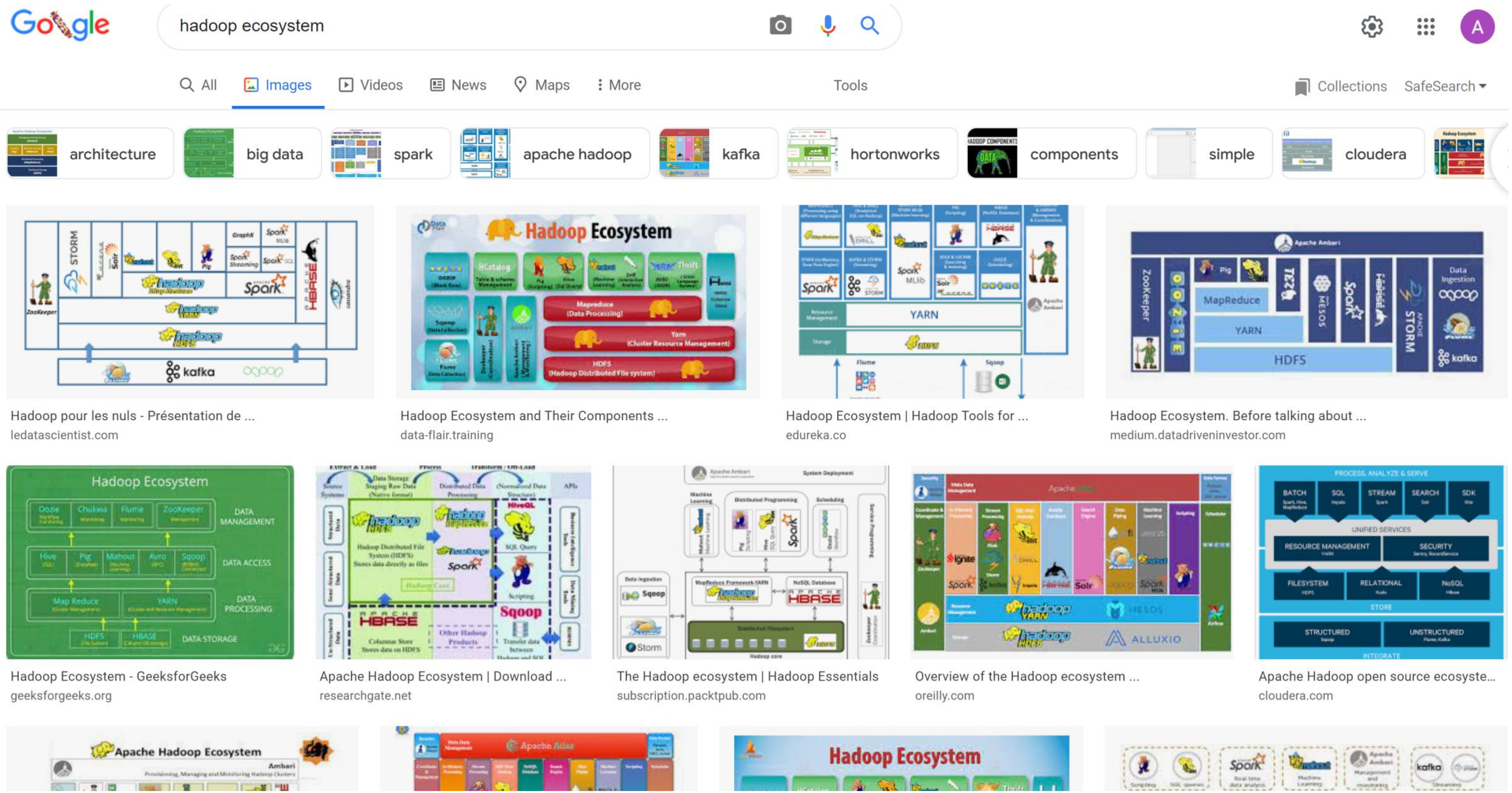
Installed Hadoop Distribution

HORTONWORKS®

How Softwares evolve to handle all this ?

Which ones to use today?

Hadoop Ecosystem « Explosion »



hadoop ecosystem

Hadoop pour les nuls - Présentation de ...
ledatascientist.com

Hadoop Ecosystem and Their Components ...
data-flair.training

Hadoop Ecosystem | Hadoop Tools for ...
edureka.co

Hadoop Ecosystem. Before talking about ...
medium.datadriveninvestor.com

Hadoop Ecosystem - GeeksforGeeks
geeksforgeeks.org

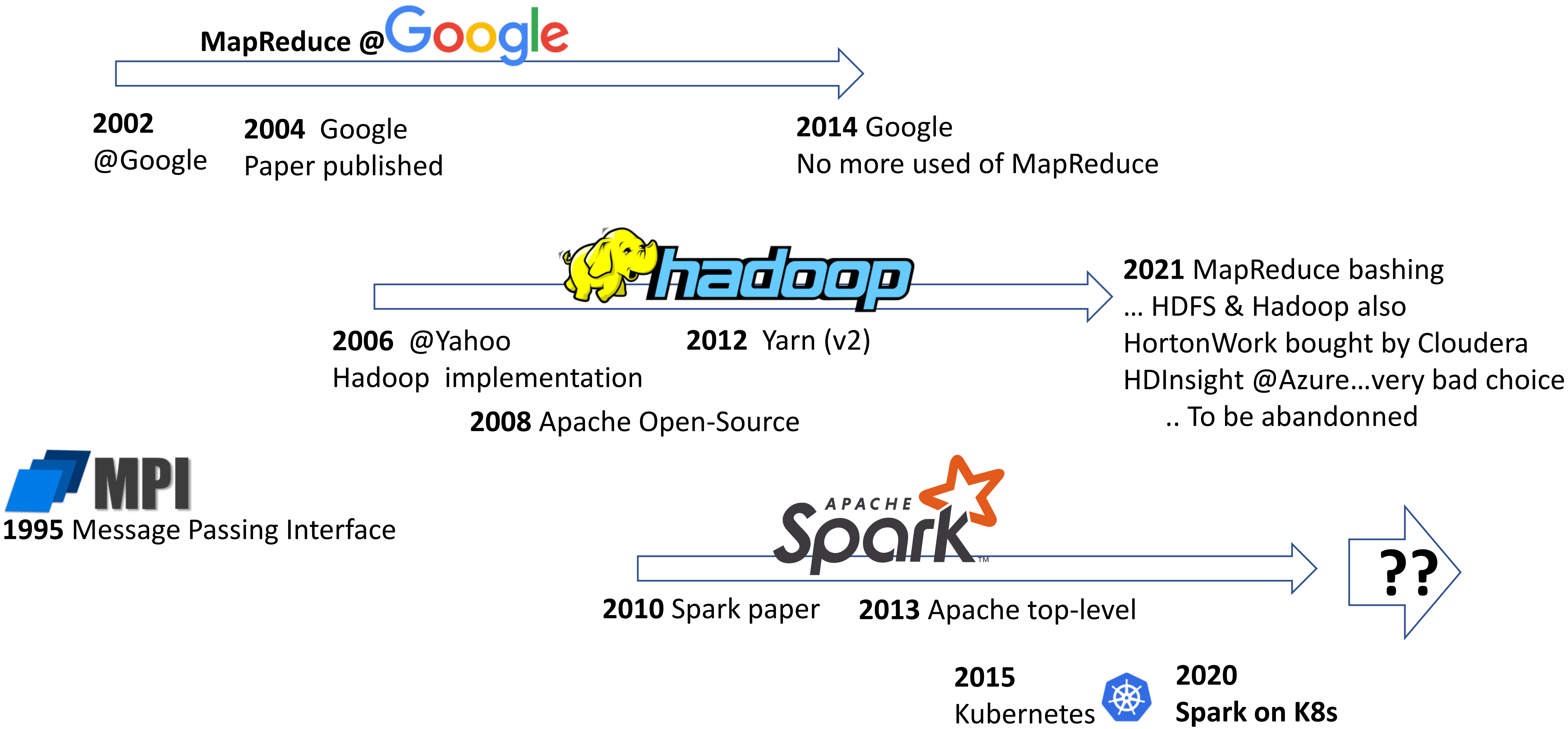
Apache Hadoop Ecosystem | Download ...
researchgate.net

The Hadoop ecosystem | Hadoop Essentials
subscription.packtpub.com

Overview of the Hadoop ecosystem ...
oreilly.com

Apache Hadoop open source ecosyste...
cloudera.com

Spark (Recent) History & Ancestors



MapReduce @Google

2002
@Google

2004 Google
Paper published

2014 Google
No more used of MapReduce

2006 @Yahoo
Hadoop implementation

2012 Yarn (v2)

2008 Apache Open-Source

2021 MapReduce bashing
… HDFS & Hadoop also
HortonWork bought by Cloudera
HDInsight @Azure…very bad choice
.. To be abandonned

1995 Message Passing Interface

2010 Spark paper

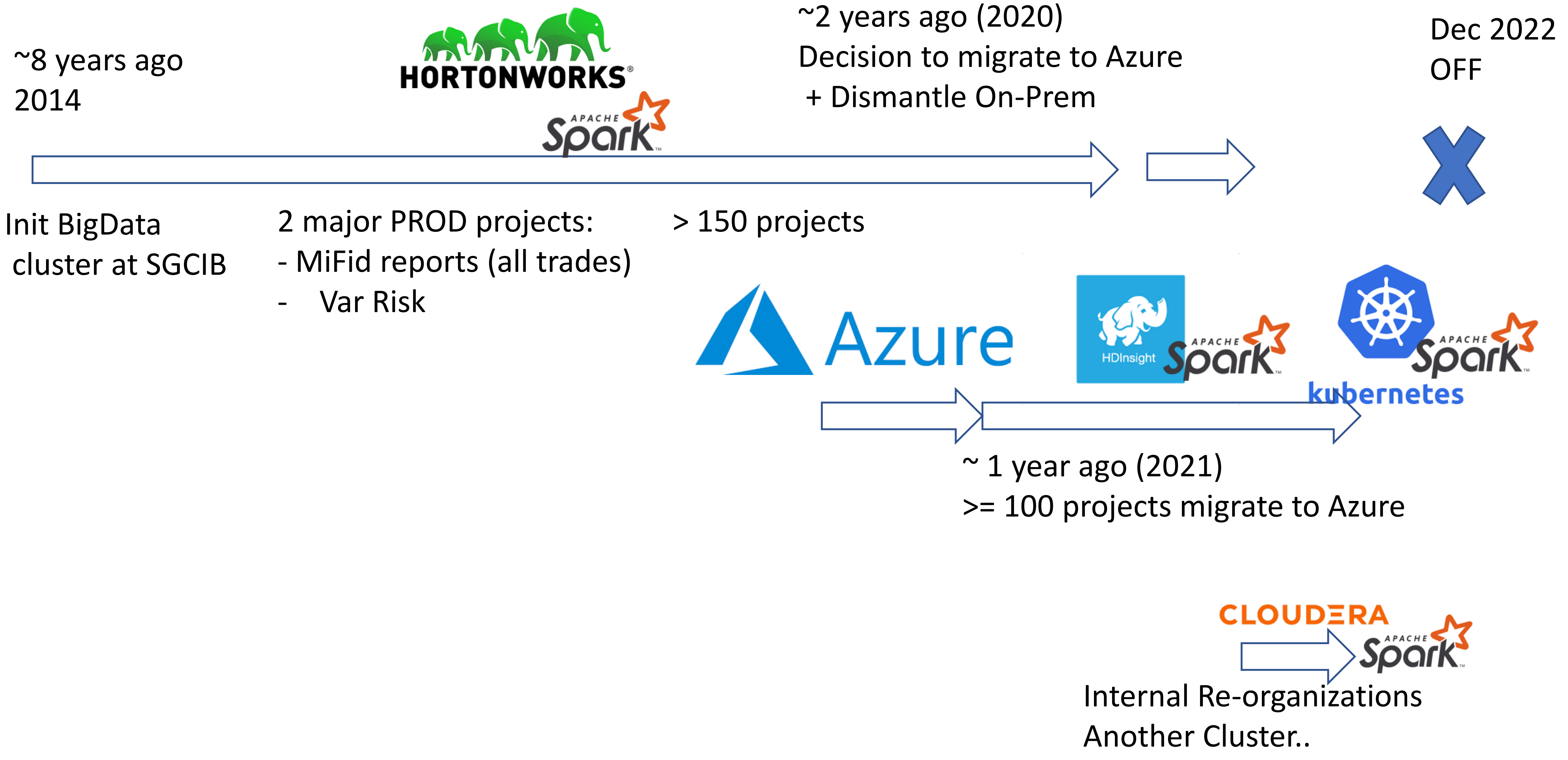
2013 Apache top-level

??

2015
Kubernetes

2020
Spark on K8s

Datalake History At SG



~8 years ago
2014

Init BigData
cluster at SGCIB

2 major PROD projects:
- MiFid reports (all trades)
- Var Risk

> 150 projects

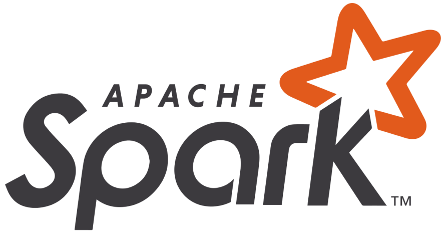
~2 years ago (2020)
Decision to migrate to Azure
+ Dismantle On-Prem

Dec 2022
OFF

~ 1 year ago (2021)
>= 100 projects migrate to Azure

Internal Re-organizations
Another Cluster..

Point in common
Despite all changes



MapReduce @Yahoo = Hadoop .. 2006

Constraint
=>
Architecture
Choice

Commodity Hardwares (datacenters):

Only HDD + RAM

Data Locality : co-host Storage near Compute

use RAM to cache

avoid network

APACHE
hadoop

Think different?

↑

2006

↑

2020

MapReduce -> Spark + other changes

Constraint

=>

Architecture

Choice

Faster Disk (SSD)

Faster Network

Compute != Storage

Cloud

APACHE Spark™

kubernetes

APACHE hadoop

2006

amazon web services

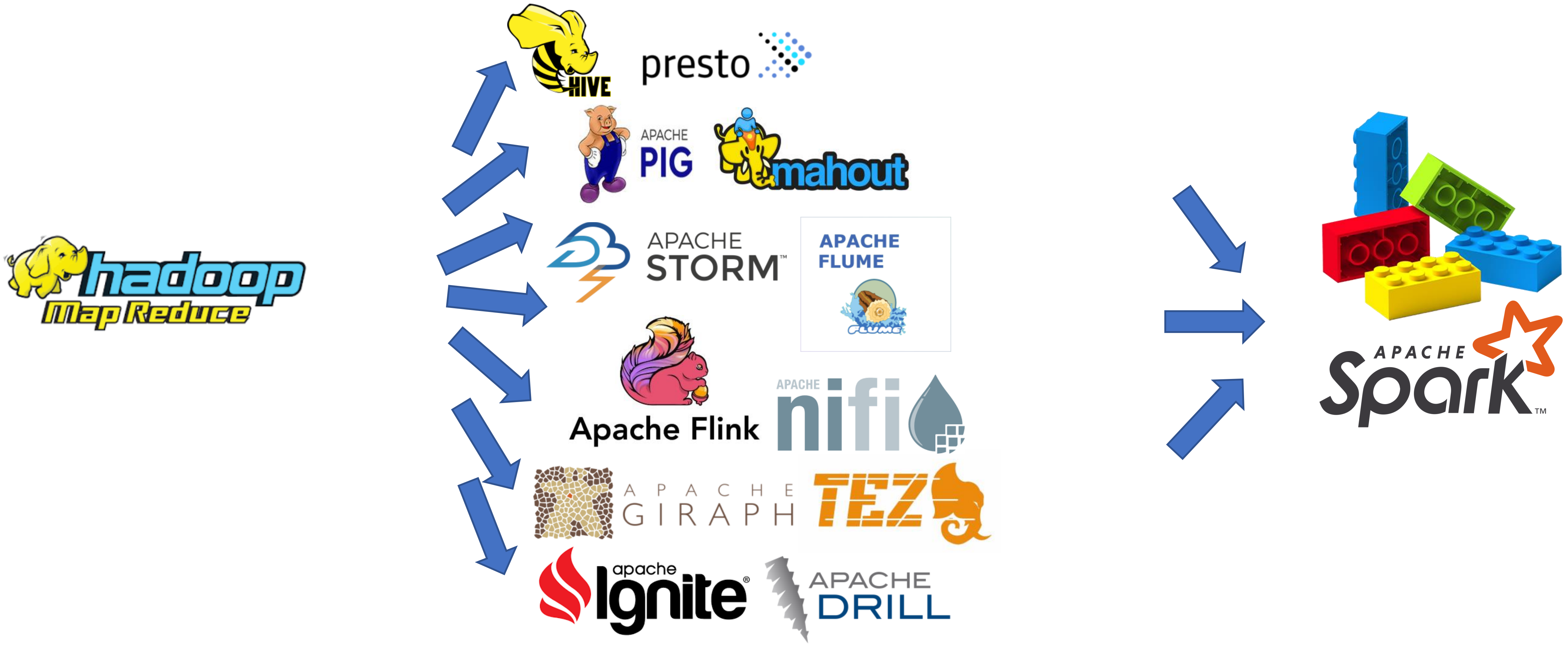
Microsoft Azure

Google Cloud

Think different?

2022

Simple => Many Specific Systems => Unified



« **Simple** » ecosystem
(verbose inneficient &
complex java code)

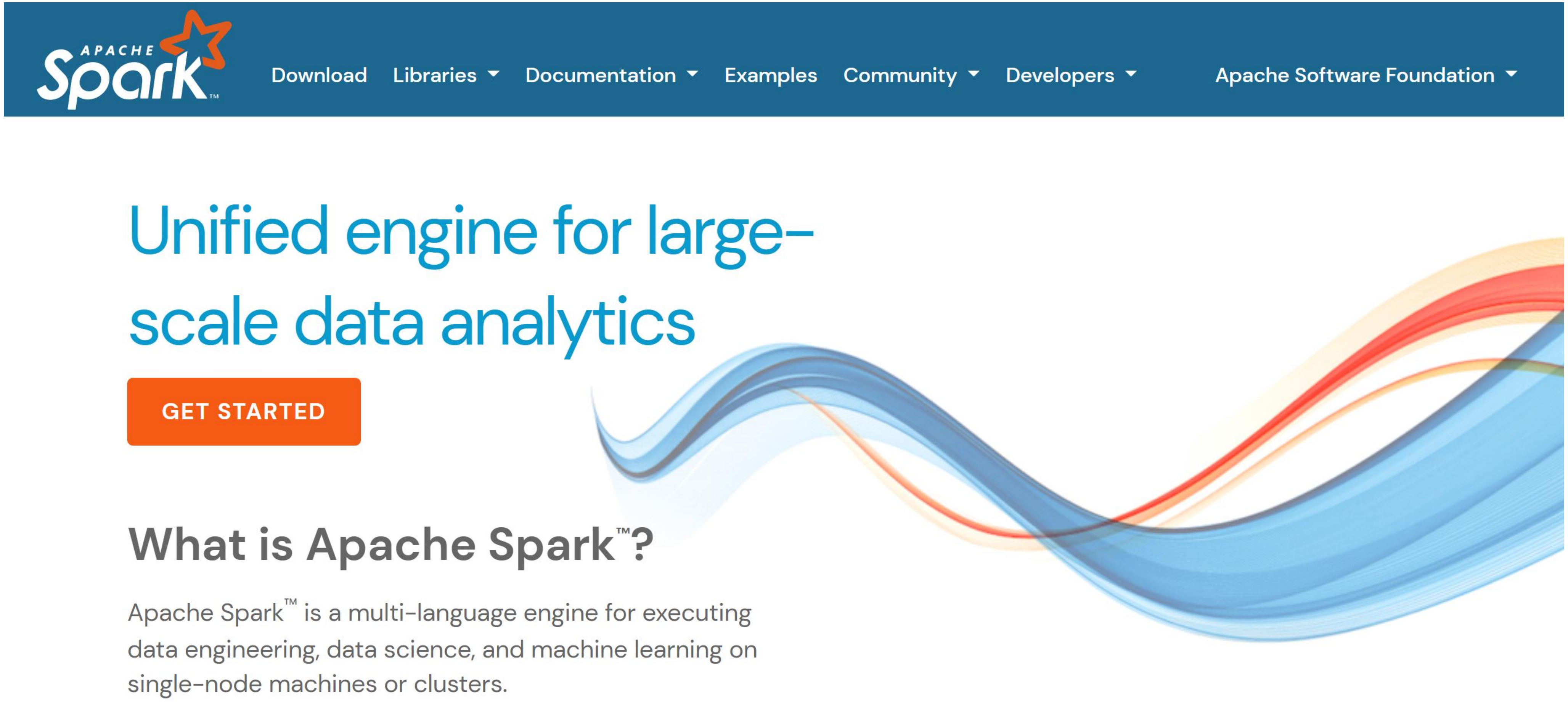
« **Bazard** » ecosystem
(**Too MANY TOO** SPECIFIC
redundant, complexes)

“Unified” ecosystem
Simple
+ extensible modules

At The end, Only 1 will remain
(French TV Game: Koh-Lanta)

APACHE
Spark
TM

Spark = « **Unified Engine** »

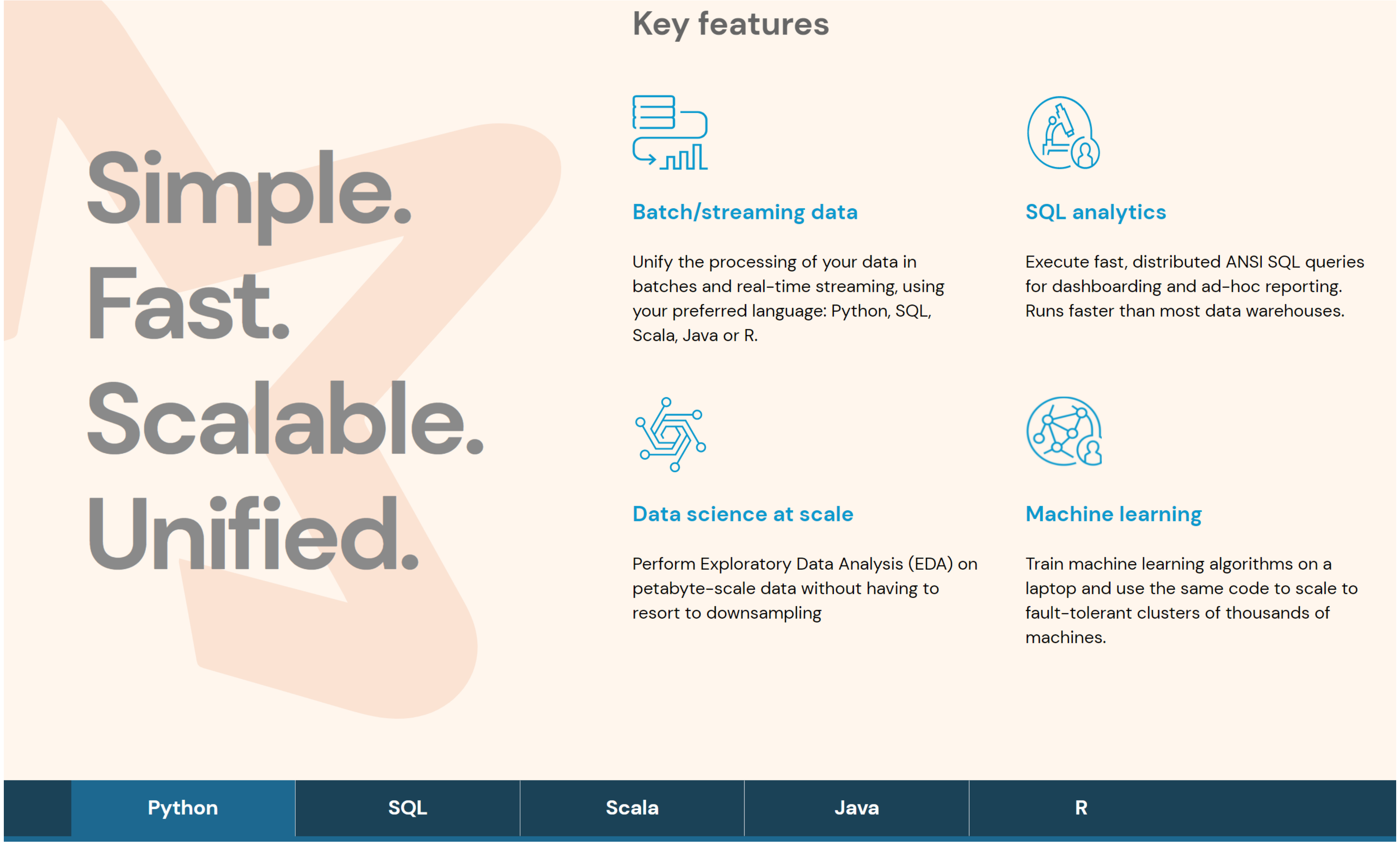


Unified engine for large-scale data analytics

What is Apache Spark™?

Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

Multi Purposes – Multi Langages



Simple.
Fast.
Scalable.
Unified.

Key features

Batch/streaming data

Unify the processing of your data in batches and real-time streaming, using your preferred language: Python, SQL, Scala, Java or R.

SQL analytics

Execute fast, distributed ANSI SQL queries for dashboarding and ad-hoc reporting. Runs faster than most data warehouses.

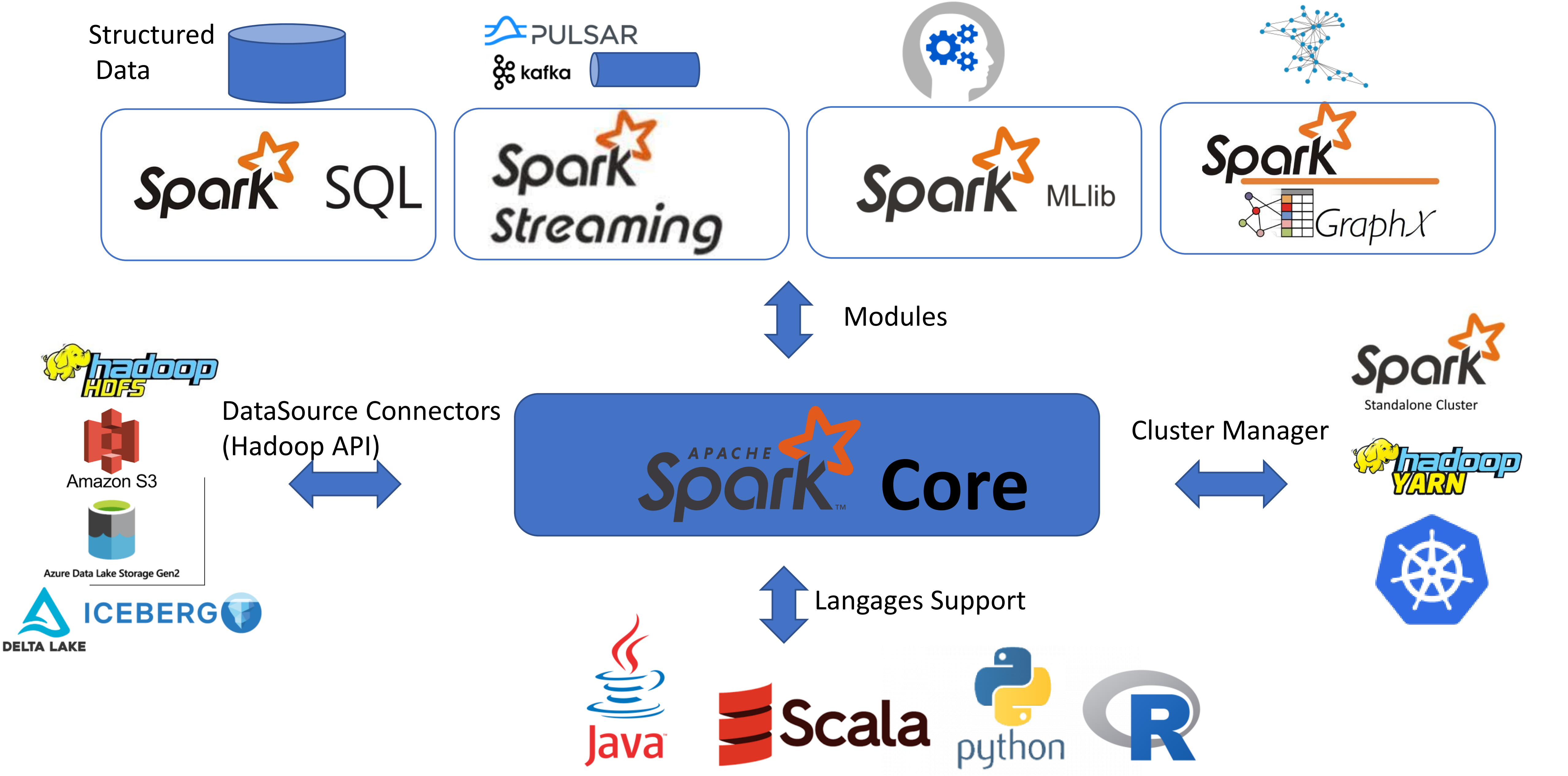
Data science at scale

Perform Exploratory Data Analysis (EDA) on petabyte-scale data without having to resort to downsampling

Machine learning

Train machine learning algorithms on a laptop and use the same code to scale to fault-tolerant clusters of thousands of machines.

Spark-Core + …



Structured Data

Modules

DataSource Connectors (Hadoop API)

Cluster Manager

Langages Support

Outline

- What is BigData ?
  - Order of Magnitudes for « Big »
  - History
  - Evolution of Softwares, Spark
- Description of a Datalake
  - Content, data feeding
  - How it is organized
  - Who uses it
- Example of Data-Processing
  - RAW to LAKE, Reports
- Change Storage-Compute, Evolution to Cloud

Not Only Machines

Tower of Babel

מִגְדַּל בָּבֶל

The Tower of Babel by Pieter Bruegel the Elder (1563)

General information

Type	Tower
Location	Babylon
Height	See § Height

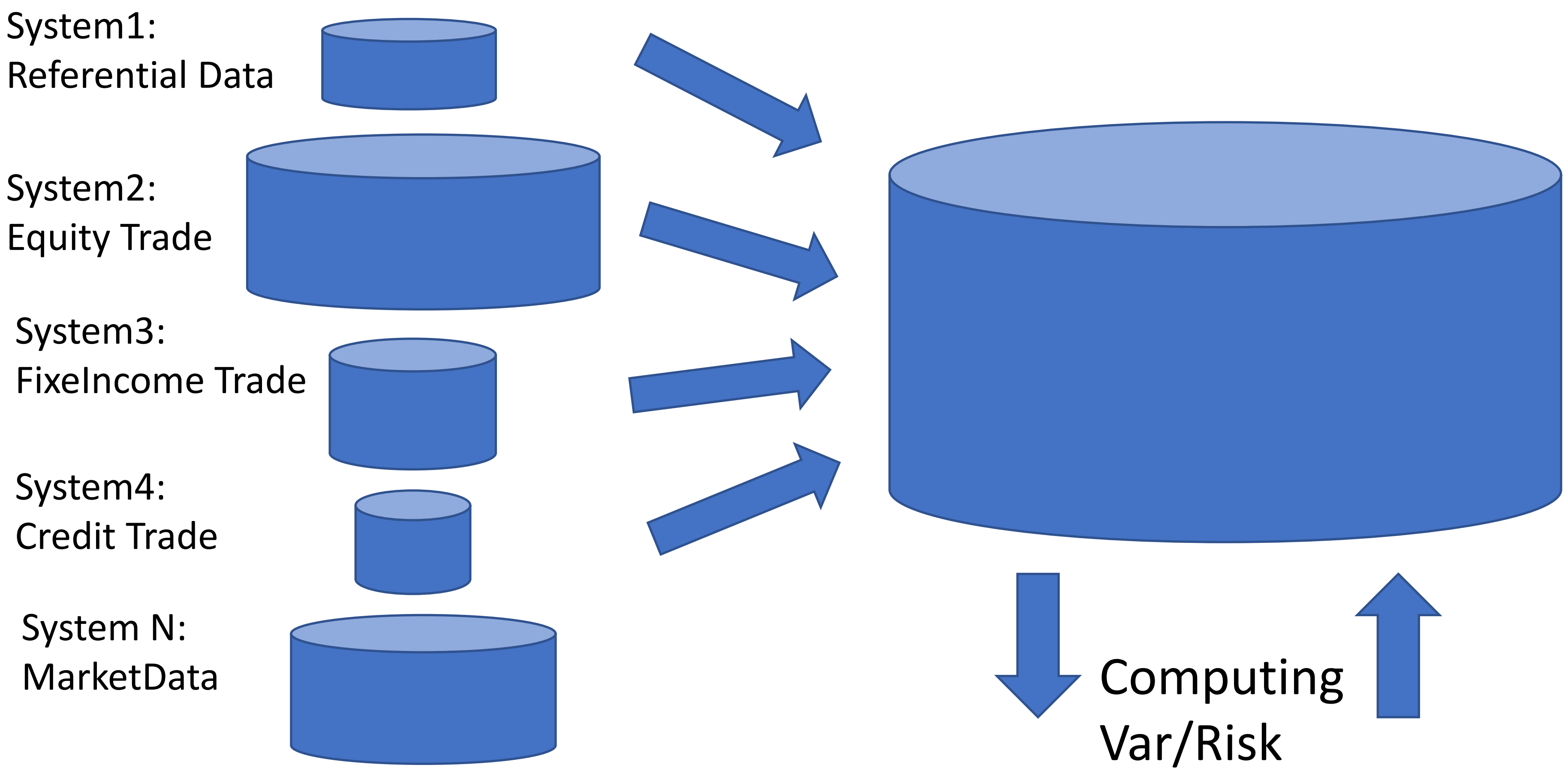
~ 200 different Teams / 10 departments
… for 1000 developpers / data engineers / users

In Paris.. But not Only (Bangalore, Montreal, London, New-York, ..)

On ~6000 tables,
described in 1500 DataSets

Lot of Financial / Functional aspects

Feeding Data Inputs to Datalake: aggregating >= 150 Systems



Feeding Data / Streaming

BigData at SG is mostly « uploading daily files »
then launching « daily batches »

Many Trade extractions are Real-Time Streaming
… but aggregated in Files per Hours / per Day

Var Computing during night … Huge streaming of events
from dedicated pricing cluster (GPUs)

Feeding Types & Volumetry

Daily batches: upload Files

Streaming then converted to Files

Teras/day
(millions of files)

Streaming to Key-Values Databases

Billions events/day

Peeks to 4000 events/seconds
on each 100 x servers
During 4hours nightly Var batches

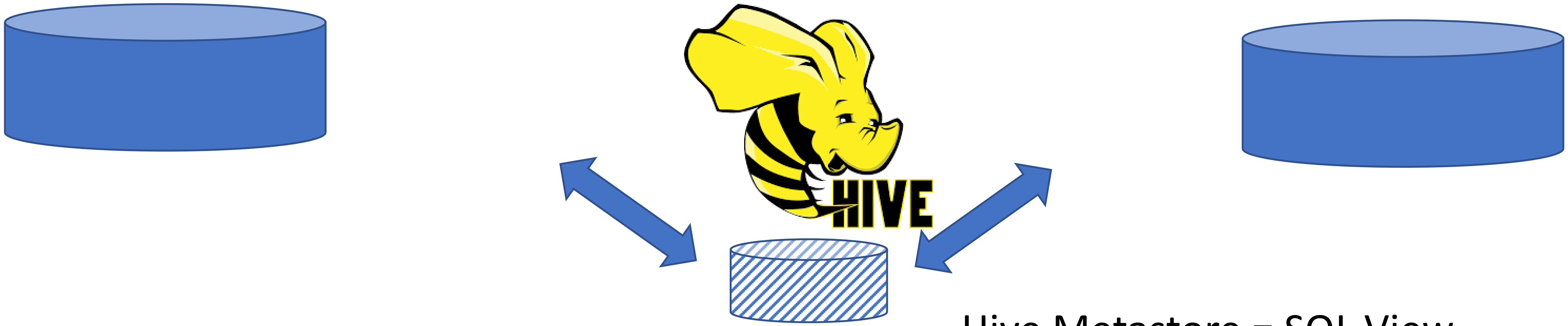
Content of a Datalake

FileSystem
= Directories + Files

For batches

Key-Value database

FAST get/put access by single key



Hive Metastore = SQL View
« Table » abstraction for HDFS / Hbase

Storage = Files and Directories

How would you organize

1 Billions Files

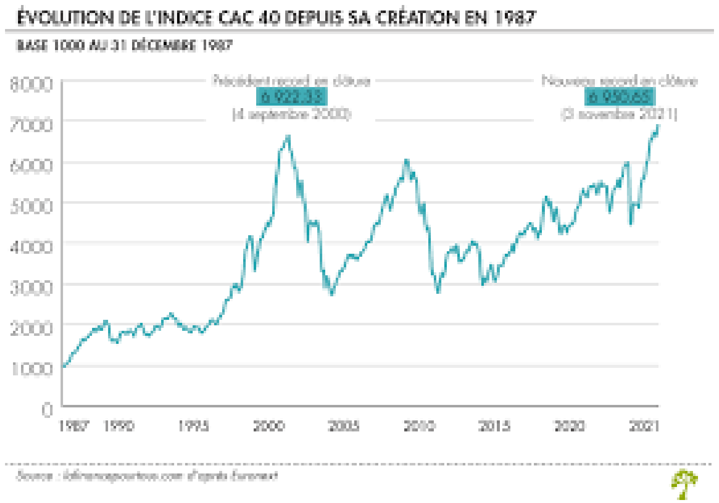
in **50 Millions Directories** ?

How do you ensure data respect directories organization ?

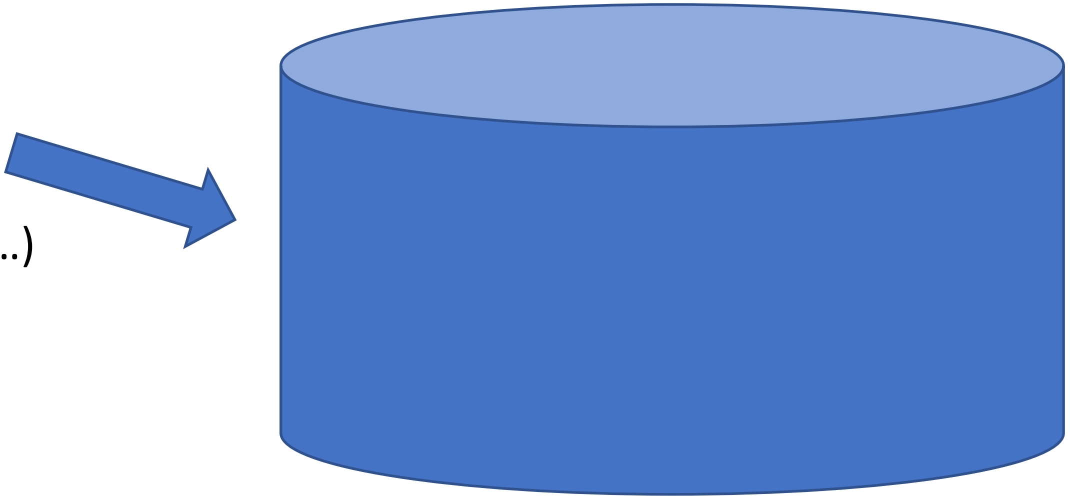
Try Scanning Directories ?

A Very « Simple » Specific Sub-System

« Tick » per seconds
From Market Exchange Places
(Bourse Paris, London Stock Exchange, …)



Source : lafinancepourtous.com d'après Euronext



1 binary file per Day/Place/Instrument
… 2M files per day
… 40 Giga per day
… since 22 years … 25 Billions Files / Petas

Volume, Velocity … but not Variety
(NO spark, No Hadoop)

Governance
Data Directories Organization
Read-Write Permissions, Quota, RDPG …

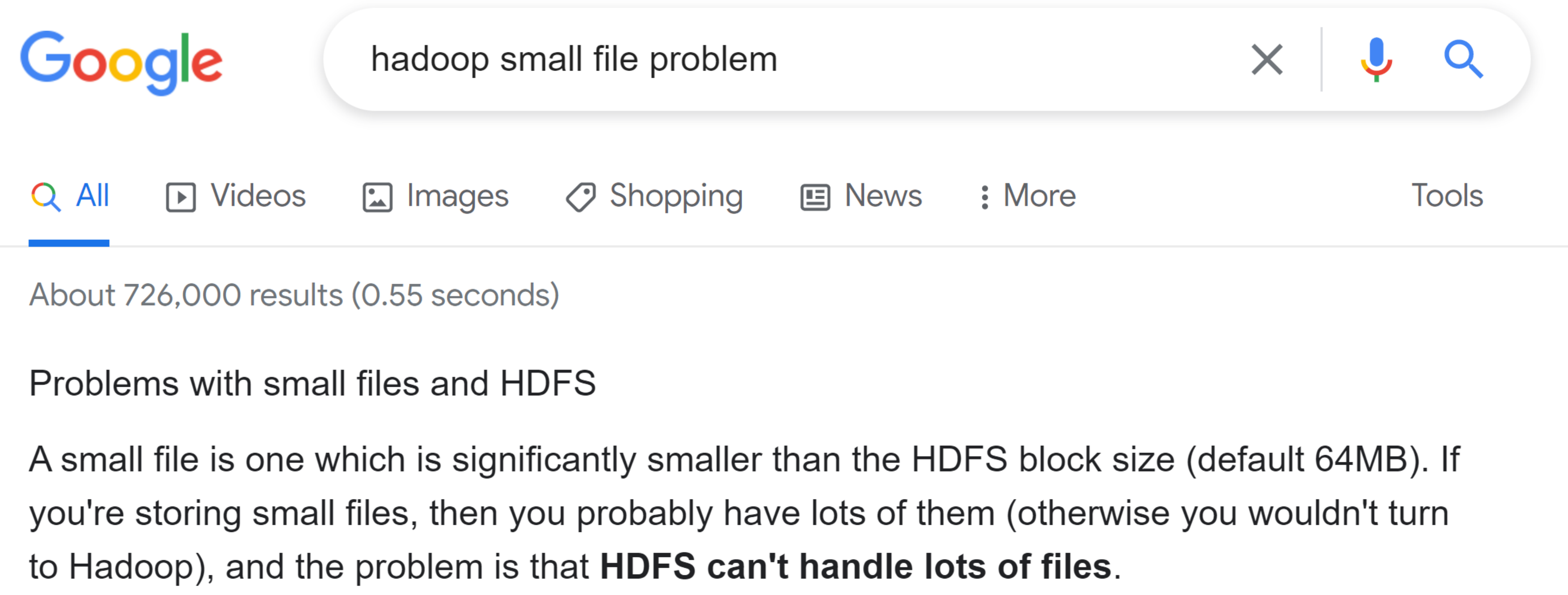
How would you organize Read-Write Permissions on 6000 tables x 1000 jobs ?

Project ask access
AND describe « Datasets »
(data owner)
=> Validation by DataOffice
=> Create domain sub-dirs
   + granted recursive
      WRITE permission

```
/ <root HDFS>
 /env
   / {raw|lake}
     /dir  =by Team
       /subDir   =by domain
         /sub-subDir=Hive table
              /hive partitions
                   /*.parquet
```

Project ask for a « Usage » Job
Explain Inputs/Output
=> Validation by DataOffice
=> Grant recursive
      READ permission for inputs

Hadoop hates too many Small Files



hadoop small file problem

About 726,000 results (0.55 seconds)

Problems with small files and HDFS

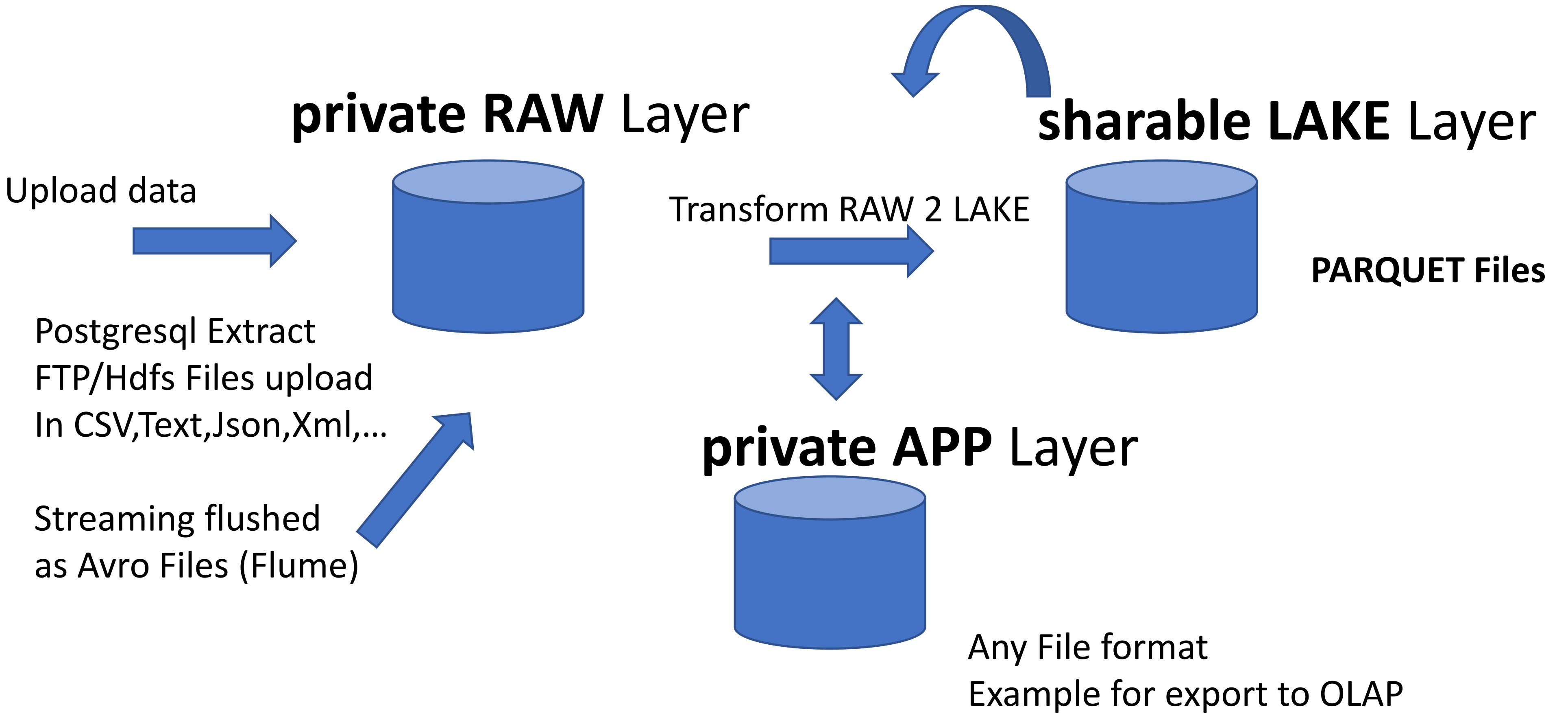
A small file is one which is significantly smaller than the HDFS block size (default 64MB). If you're storing small files, then you probably have lots of them (otherwise you wouldn't turn to Hadoop), and the problem is that **HDFS can't handle lots of files**.

BIG Files ? More than Giga ?
YES … BUT PARQUET Partitioned

Doing BigData = using Parquet Files (correctly with APACHE Spark™)

End Of Story.

LAKE = Parquet
Starting from RAW: Postgresql or {CSV|Json|Avro..}
STAGING Dirs



private RAW Layer

Upload data

Postgresql Extract
FTP/Hdfs Files upload
In CSV,Text,Json,Xml,…

Streaming flushed
as Avro Files (Flume)

Transform RAW 2 LAKE

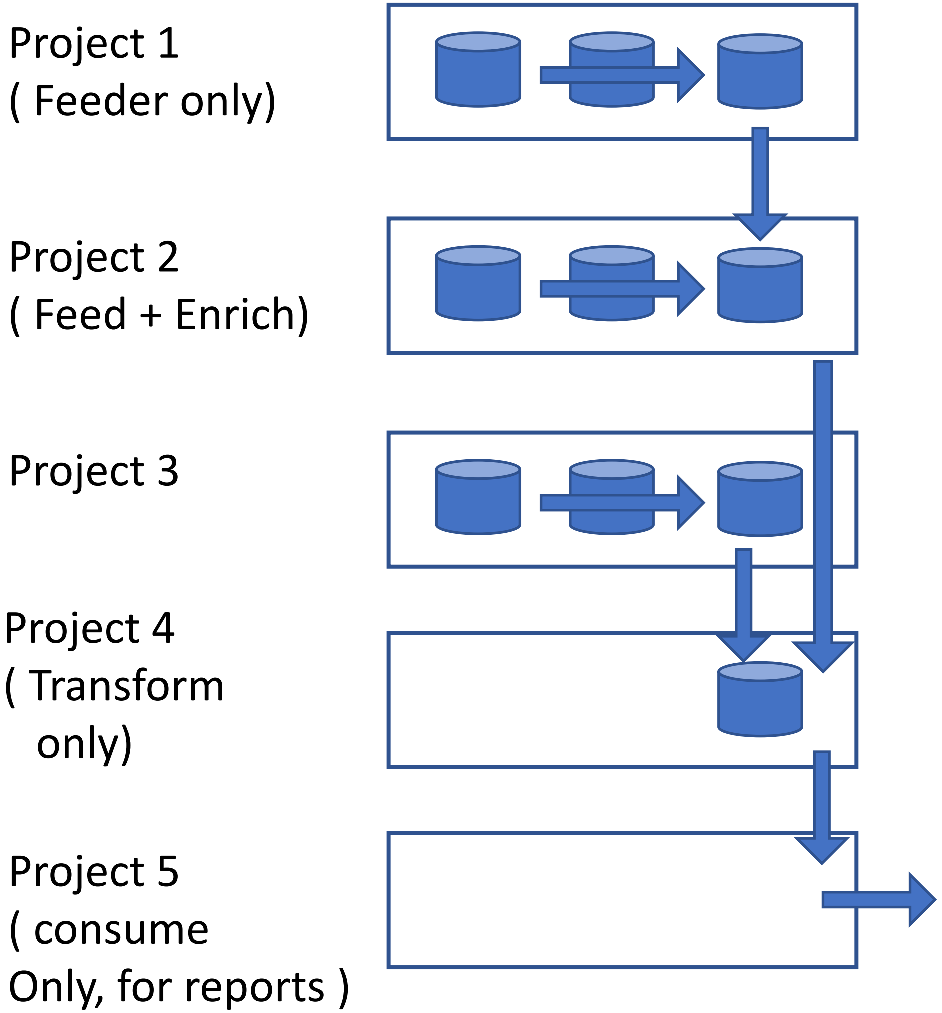
sharable LAKE Layer

PARQUET Files

private APP Layer

Any File format
Example for export to OLAP

Feeding – Transforming – Consuming Data



Project 1
(Feeder only)

Project 2
(Feed + Enrich)

Project 3

Project 4
(Transform
only)

Project 5
(consume
Only, for reports)

Every project is granted
3 Read-Write Databases: RAW, APP, LAKE

LAKE are sharable
… designed to be efficiently RE-read after (for analytical)

Every project can request Read access on others LAKE
(after validation by Data-Office)

A Dataset is writable by only 1 Project (the owner)

Big Processing

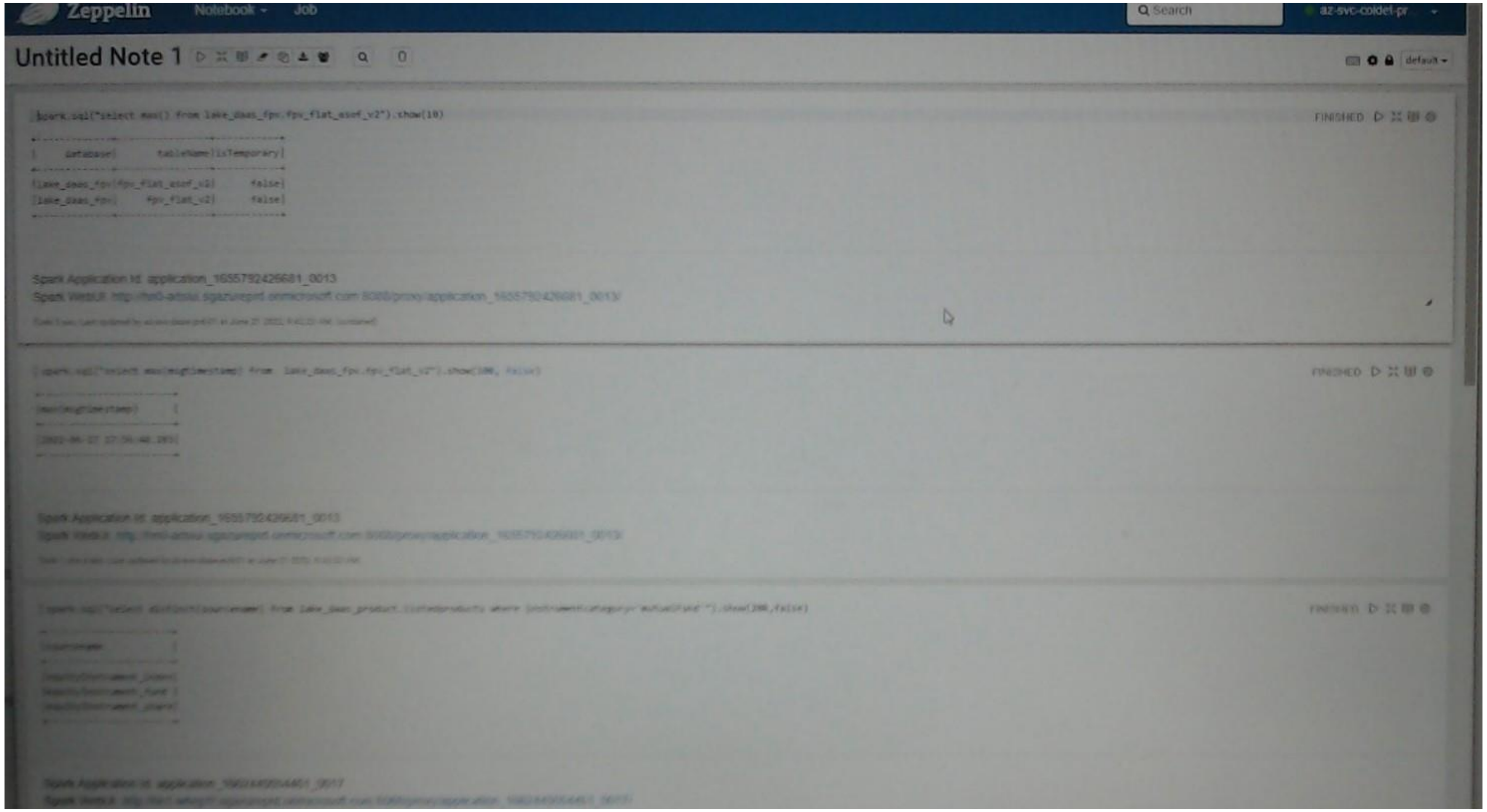
>= 1000 periodic jobs running daily / or every hours

Some processing take >= 15 Hours

Some processing are distributed on more than 200 yarn containers

Some take more than 1000 Go of RAM

Few Interactive Queries Processing screenshot Zeppelin



```
spark.sql("select max() from lake_daas_fpv.fpv_flat_asof_v2").show(10)

+--------------+-----------------+-----------+
|      database|        tableName|isTemporary|
+--------------+-----------------+-----------+
|lake_daas_fpv|fpv_flat_asof_v2|      false|
|lake_daas_fpv|      fpv_flat_v2|      false|
+--------------+-----------------+-----------+
```

Spark Application Id: application_1655792426681_0013
Spark WebUI: http://hn0-adsla.sgazurepirt.onmicrosoft.com:8088/proxy/application_1655792426681_0013/

[illegible]

```
spark.sql("select max(msgtimestamp) from  lake_daas_fpv.fpv_flat_v2").show(100, false)

+-----------------------+
|max(msgtimestamp)      |
+-----------------------+
|2022-06-27 17:56:48.283|
+-----------------------+
```

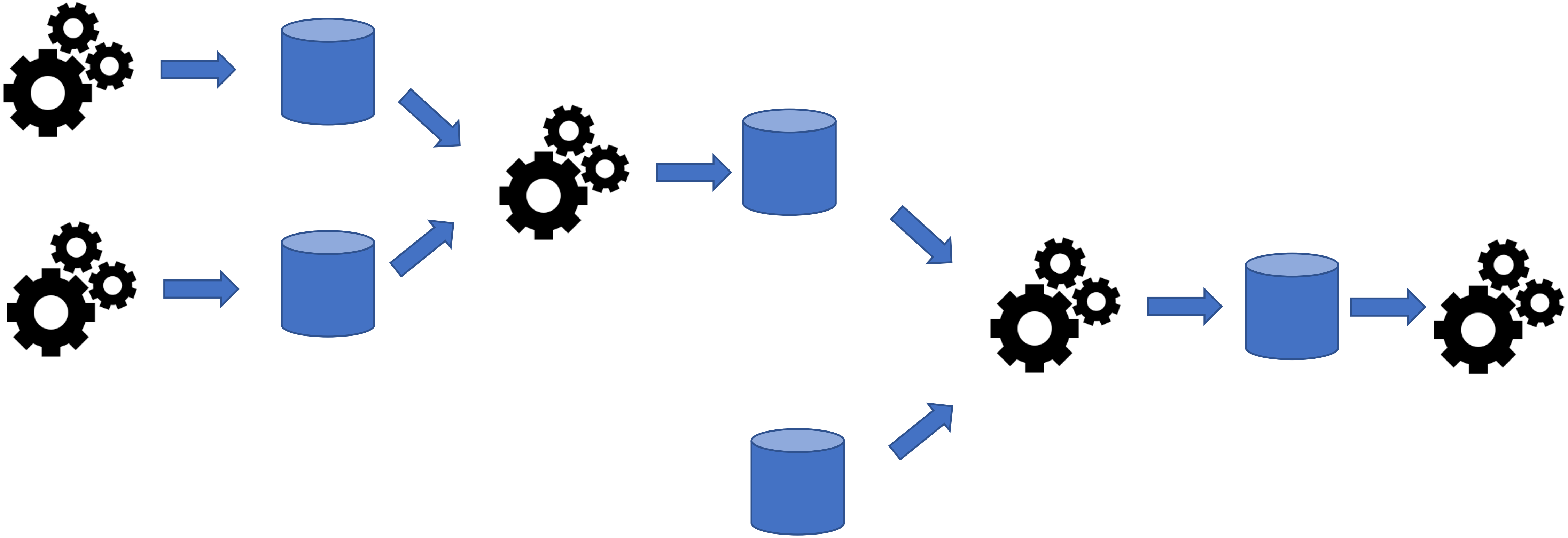
Spark Application Id: application_1655792426681_0013
Spark WebUI: http://hn0-adsla.sgazurepirt.onmicrosoft.com:8088/proxy/application_1655792426681_0013/

[illegible]

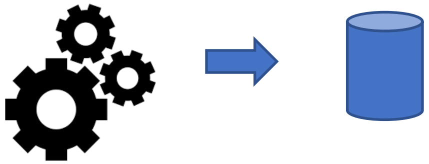
```
spark.sql("select distinct(sourcename) from lake_daas_product.listedproducts where instrumentcategory='mutualfund' ").show(100,false)

+--------------------+
|sourcename          |
+--------------------+
|[illegible]         |
|[illegible]         |
|[illegible]         |
+--------------------+
```

Spark Application Id: application_1582449554451_0017
Spark WebUI: http://hn0-[illegible].sgazurepirt.onmicrosoft.com:8088/proxy/application_1582449554451_0017/

Orchestration / DataLineage
Data and Workflows are linked



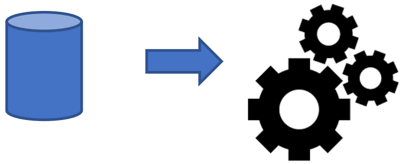
Read-Write Permissions from DataOffice



Dataset « Feeding » declaration:

= exclusive Write Permission :   Project -> canWrite -> DB

For validation by DataOffice, DataSet must be described :
confidentiality, source, business owner, quality, RDPG legal

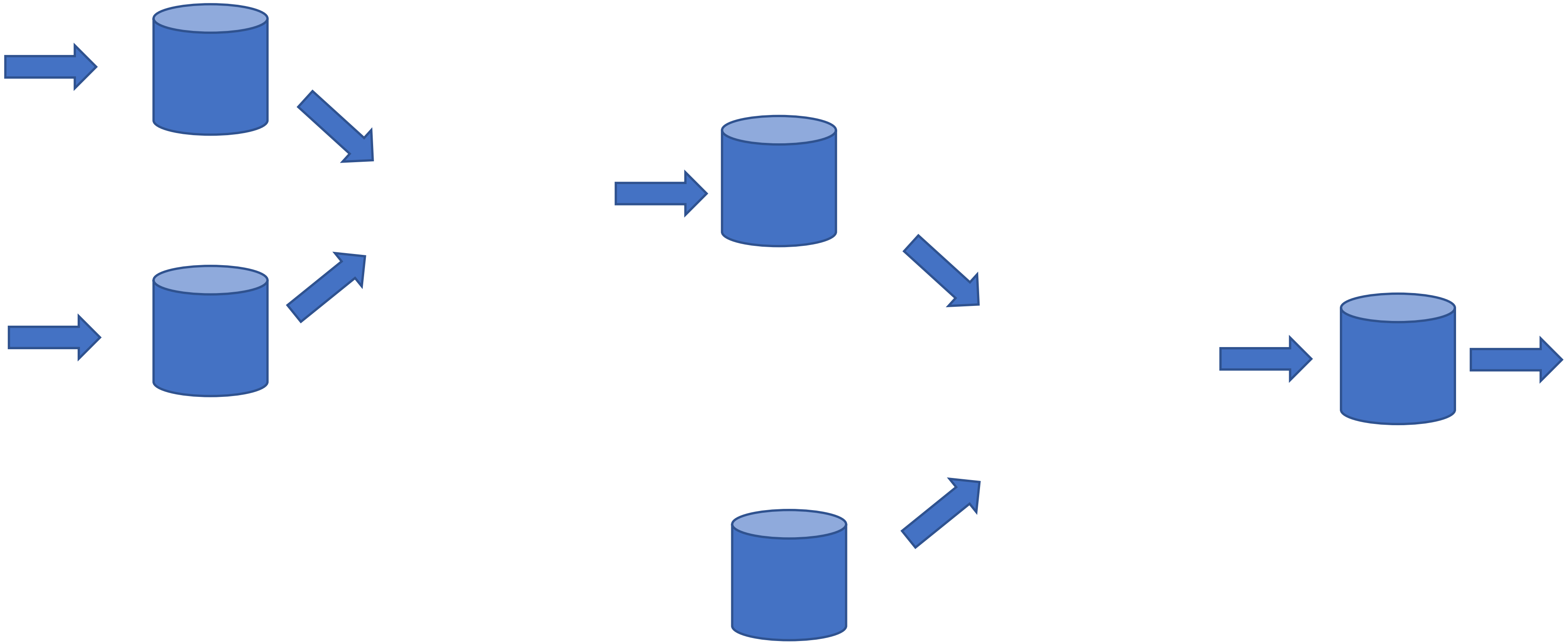


Consumer declaration:

= authorization to read : Project -> canRead -> DB
… for a specific Usage

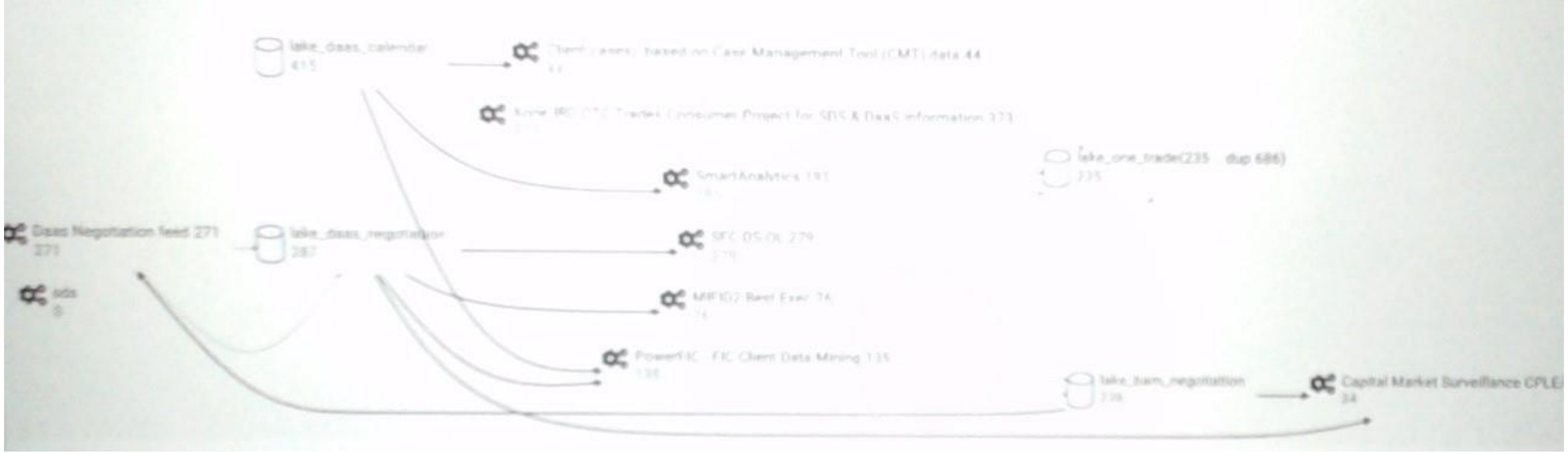
For validation by DataOffice, use-case must be described,
usage must be « legitimate » , RDI legal

DataLineage
Data depends on other Sources Data

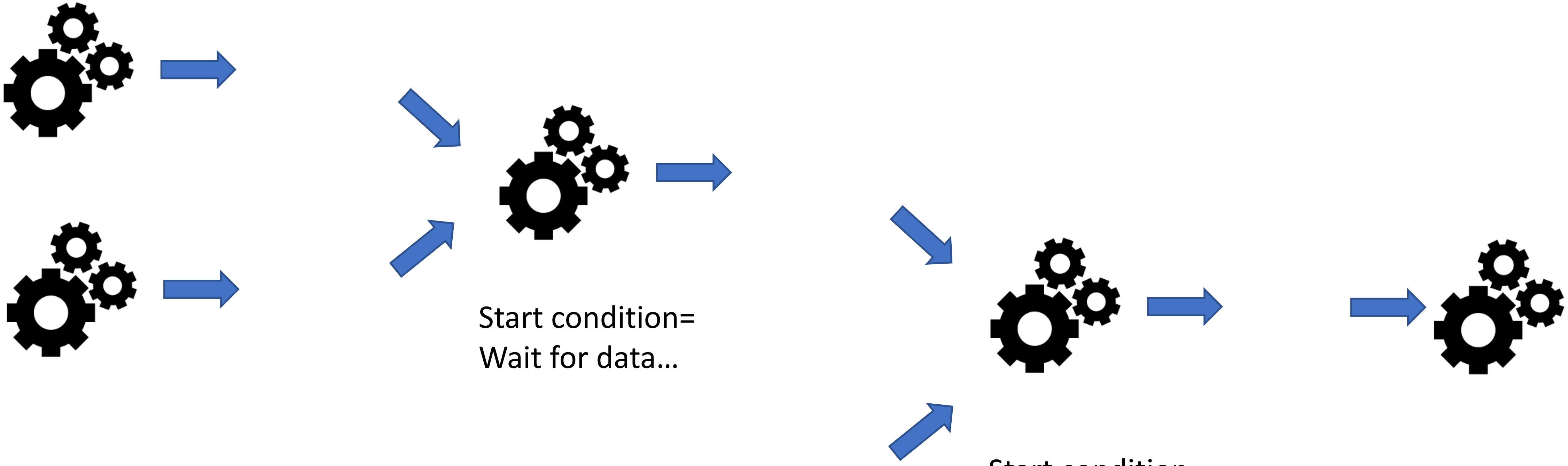


Screenshots (Fuzzy.. On purpose)
Read/Write Dependencies Hive Database / Job…

Zooming on 4 out of 250 Hive databases


lake_daas_calendar
415
Client cases - based on Case Management Tool (CMT) data 44
44
[illegible] Trades Consumer Project for SDS & DaaS information 323
SmartAnalytics 191
191
lake_one_trade(235 - dup 686)
235
Daas Negotiation feed 271
271
lake_daas_negotiation
287
SFC DS OL 279
279
sds
8
MIFID2 Best Exec 76
76
PowerFIC - FIC Client Data Mining 135
135
lake_bam_negotiation
238
Capital Market Surveillance CPLE
34

Orchestration
Start process2 after process1 finished

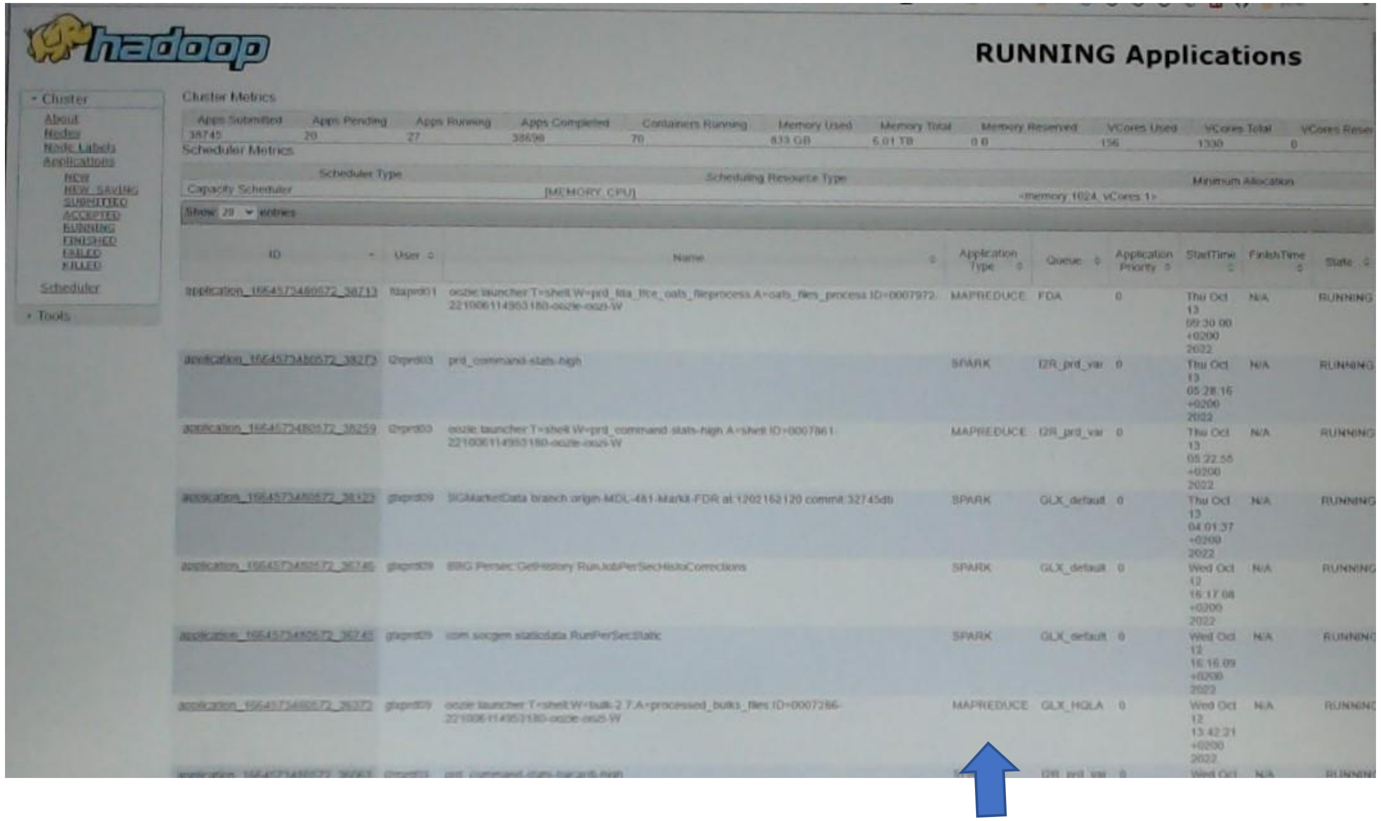


Start condition=
CRON 1 * * * * * *

Start condition=
Wait for data…

Start condition=
CRON 10 * * * * * *

Prod Screenshot (Fuzzy.. on purpose)
Running Yarn Applications



RUNNING Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

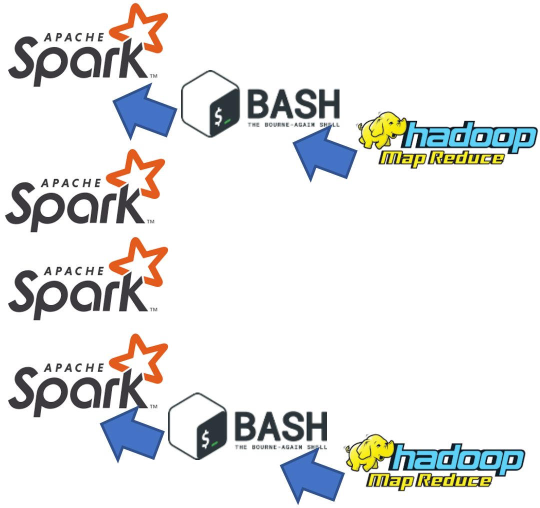
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reser
38745	20	27	38698	70	833 GB	6.01 TB	0 B	156	1330	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[MEMORY, CPU]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State
application_1664573480572_38713	fdaprd01	oozie:launcher:T=shell:W=prd_fda_fice_oats_fileprocess:A=oats_files_process:ID=0007972-221006114953180-oozie-oozi-W	MAPREDUCE	FDA	0	Thu Oct 13 09:30:00 +0200 2022	N/A	RUNNING
application_1664573480572_38273	l2rprd03	prd_command-stats-high	SPARK	I2R_prd_var	0	Thu Oct 13 05:28:16 +0200 2022	N/A	RUNNING
application_1664573480572_38259	l2rprd03	oozie:launcher:T=shell:W=prd_command-stats-high:A=shell:ID=0007861-221006114953180-oozie-oozi-W	MAPREDUCE	I2R_prd_var	0	Thu Oct 13 05:22:55 +0200 2022	N/A	RUNNING
application_1664573480572_38123	glxprd09	SGMarketData branch:origin-MDL-481-Markit-FDR at:1202162120 commit:32745d8	SPARK	GLX_default	0	Thu Oct 13 04:01:37 +0200 2022	N/A	RUNNING
application_1664573480572_36746	glxprd09	BBG Persec GetHistory RunJobPerSecHistoCorrections	SPARK	GLX_default	0	Wed Oct 12 16:17:08 +0200 2022	N/A	RUNNING
application_1664573480572_36745	glxprd09	com.socgen.staticdata.RunPerSecStatic	SPARK	GLX_default	0	Wed Oct 12 16:16:09 +0200 2022	N/A	RUNNING
application_1664573480572_36372	glxprd09	oozie:launcher:T=shell:W=bulk-2.7:A=processed_bulks_files:ID=0007286-221006114953180-oozie-oozi-W	MAPREDUCE	GLX_HQLA	0	Wed Oct 12 13:42:21 +0200 2022	N/A	RUNNING
application_1664573480572_36063	l2rprd03	prd_command-stats-bacardi-high	[illegible]	I2R_prd_var	0	Wed Oct	N/A	RUNNING

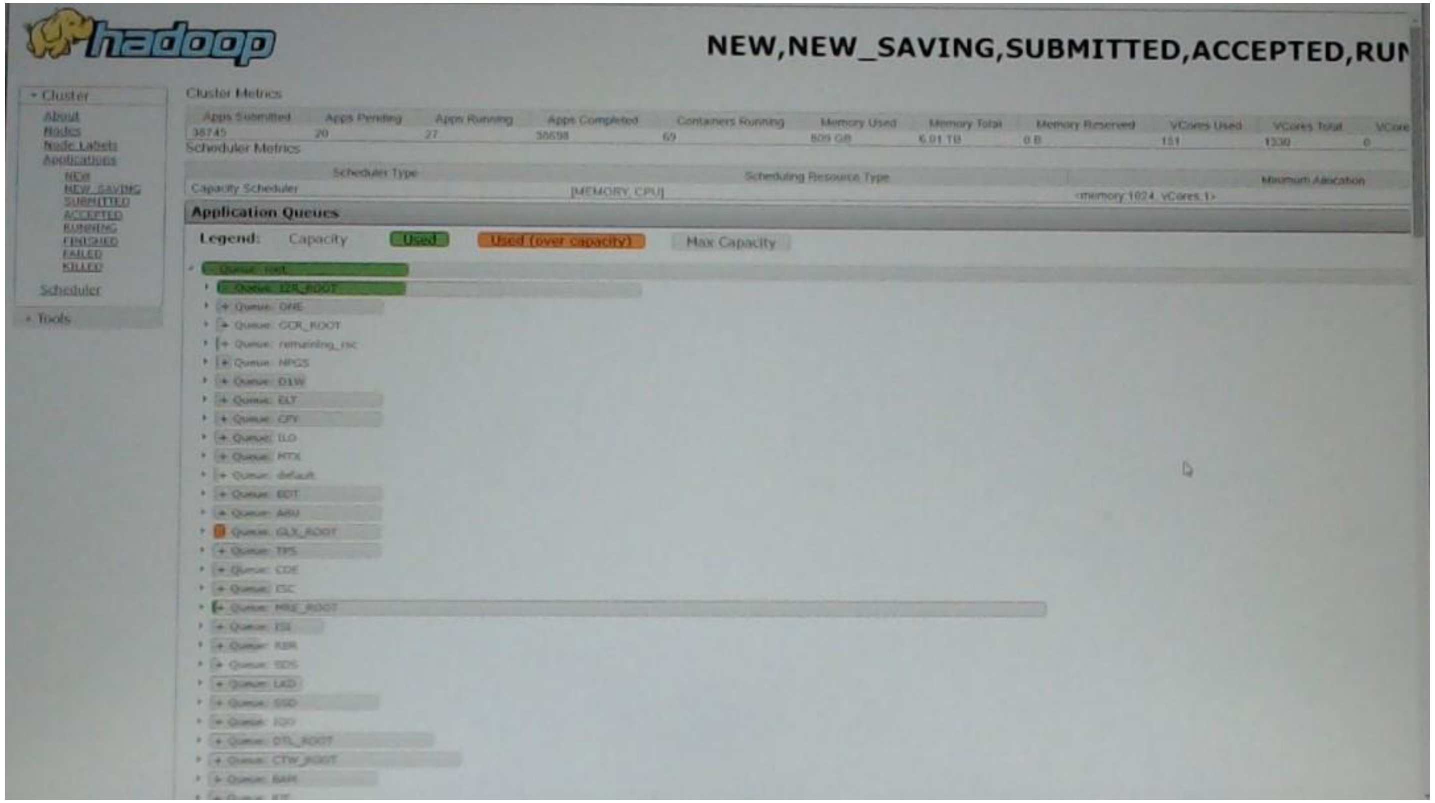


MAPREDUCE ?

… only internally to launch shell commands.. To spark

ALL others = SPARK

Prod Screenshot : Yarn Queues (Scheduler, CPU+Mem usages)



NEW,NEW_SAVING,SUBMITTED,ACCEPTED,RUN

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCore
38745	20	27	38698	69	809 GB	6.01 TB	0 B	151	1330	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[MEMORY, CPU]	<memory:1024, vCores:1>

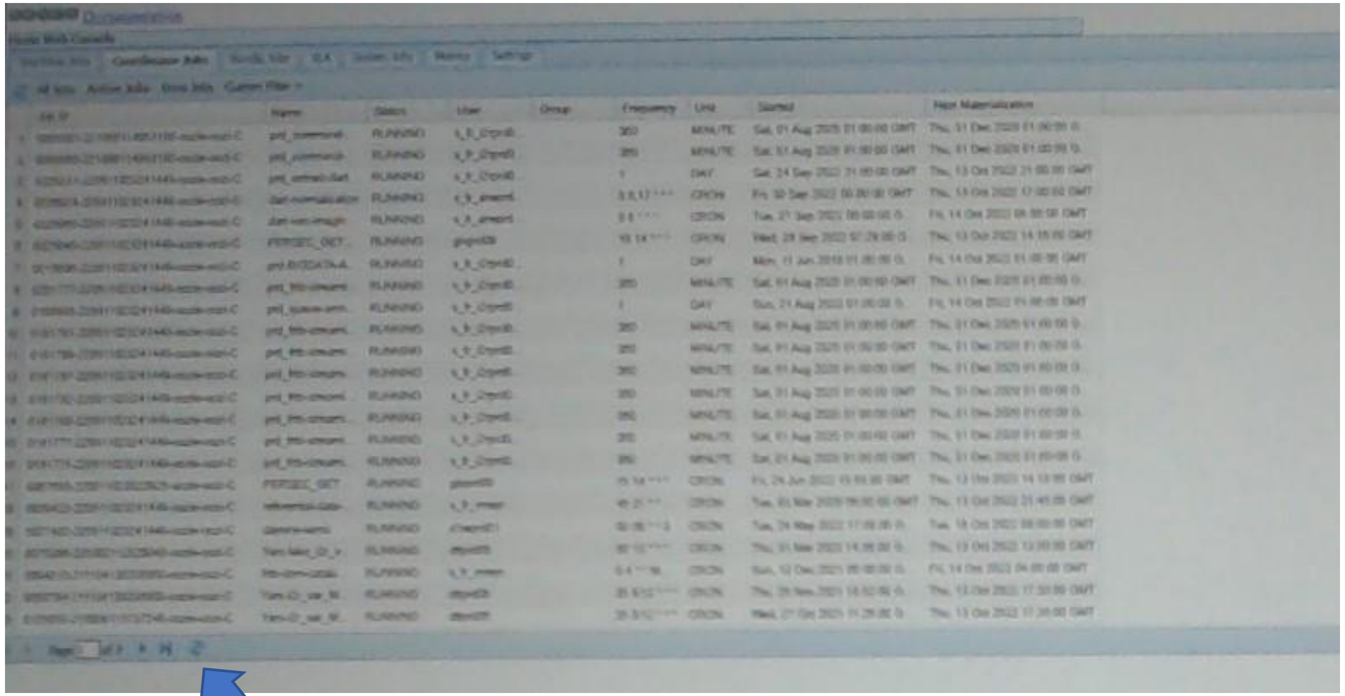
Application Queues

Legend: Capacity Used Used (over capacity) Max Capacity

- Queue: root
  - Queue: 12R_ROOT
  - Queue: ONE
  - Queue: GCR_ROOT
  - Queue: remaining_isc
  - Queue: NPGS
  - Queue: D1W
  - Queue: ELT
  - Queue: CPY
  - Queue: ILO
  - Queue: HTX
  - Queue: default
  - Queue: BDT
  - Queue: ABU
  - Queue: GLX_ROOT
  - Queue: TPS
  - Queue: CDE
  - Queue: CSC
  - Queue: MRE_ROOT
  - Queue: ESI
  - Queue: KBR
  - Queue: SDS
  - Queue: LKD
  - Queue: SSD
  - Queue: IQO
  - Queue: DTL_ROOT
  - Queue: CTW_ROOT
  - Queue: BAM
  - Queue: KIT

You see scrollbar?... Hundreds or Yarn Queues
+ dozen of sub-queues

Screenshot
Oozie (Coordinators / Workflows)

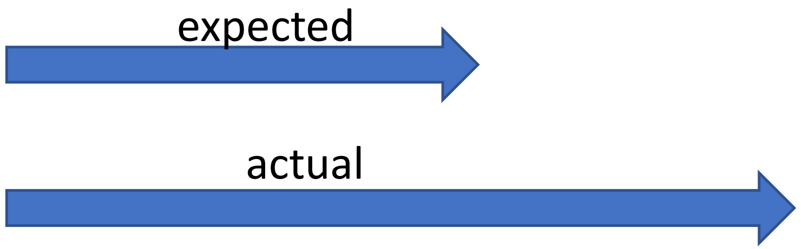


You see pagination?...
Thousands of CRON jobs

Oozie display only last 50 000 workflows
… so only 2 days of history

Orchestration « By CRON: 10 * * * * * »
Example of Possible Failures

batch1 started at 1 *** … but takes today (anormally) 15h instead of 10h

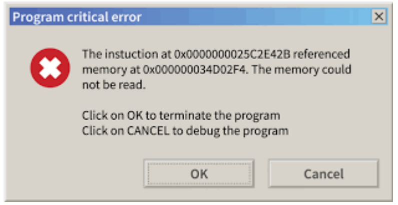


batch2 started at 12 *** …. data not yet present

➡ ∅ => NO « today » data (only yesterday)
Finished FAST, with NO data

batch2 started, during critical section of batch1 … table dropped/recreated after

➡



Program critical error

The instuction at 0x000000025C2E42B referenced memory at 0x000000034D02F4. The memory could not be read.

Click on OK to terminate the program
Click on CANCEL to debug the program

Outline

- What is BigData ?
  - Order of Magnitudes for « Big »
  - History
  - Evolution of Softwares, Spark
- Description of a Datalake
  - Content, data feeding
  - How it is organized
  - Who uses it
- Example of Data-Processing
  - RAW to LAKE, Reports
- Change Storage-Compute, Evolution to Cloud

Typical Example of adding 1 daily partition

```
$ hdfs dfs –mkdir hdfs://raw/team/domain/table/date=2022-10-12

$ hdfs dfs –put localFile hdfs://raw/team/domain/table/date=2022-10-12/

$ spark-sql –master local[1] \
       –e « ALTER TABLE raw_lake_domain.table
            ADD PARTITION ( date=‘2022-10-12’ ) »
```

Typical RAW to LAKE processing with Spark as Java code

read

```
spark.read
   .format(« csv »)
   .option(«schema », « col1 type1, … colN typeN »)
   .load(« hdfs://raw/team/domain/table/date=2022-10-12 »)
```

transform

```
  .map(row ->  transformData(row) )
```

write

```
  .repartition(3, « col1 »)
  .sortWithinPartition(« col1, col2, col3 »)
  .write
  .format(« parquet »)
  .save(« hdfs://lake/team/domain/table/date=2022-10-22 »)
  ;
```

Typical RAW to LAKE processing with Spark as SQL code

```
write        INSERT OVERWRITE
                   lake_team_domain.table
             SELECT /* +REPARTITION(col1, 3) */
                col1, col2,
transform       udf_func1(col3, col4)  as col3,
                udf_func2(col4, col5)  as col4,
                ..
read         FROM
                   raw_team_domain.table
transform    JOIN
                   lake_anotherTeam_domain.anotherTable x ON x.ID=id
read         WHERE date='2022-10-22' AND ..
write        SORT BY col1, col2, col3     -- idem sortWithinPartition
```

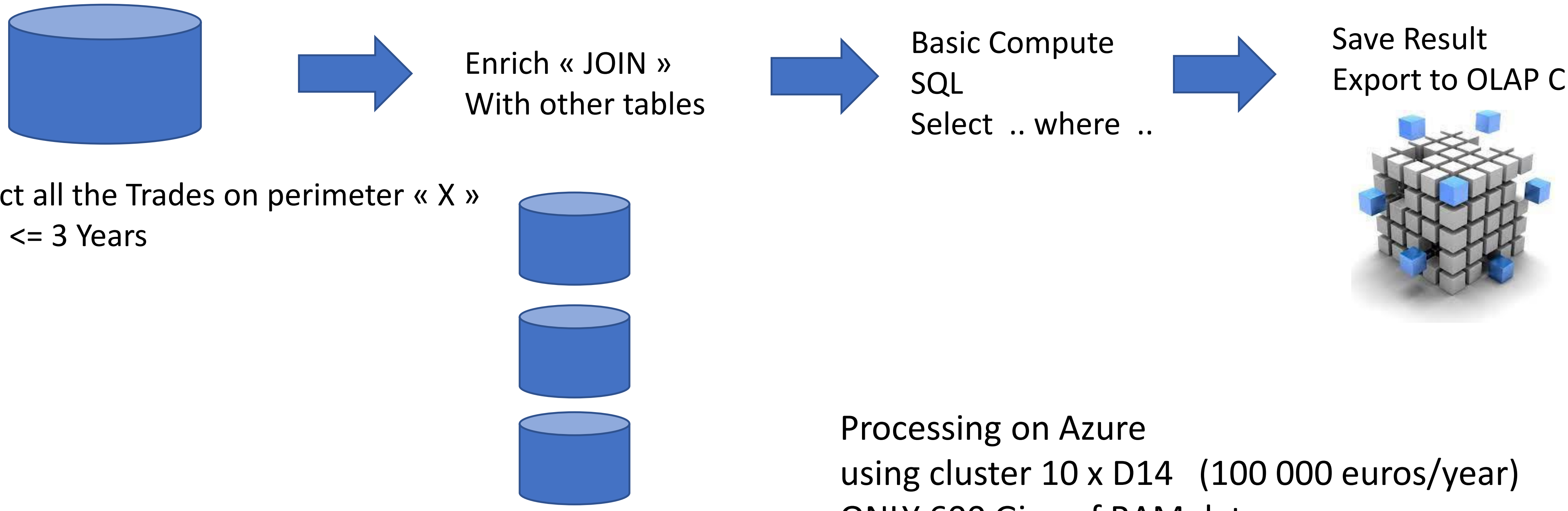
Example of LAKE Aggregation

```
INSERT OVERWRITE
       lake_team_domain.table
SELECT * FROM (
     SELECT * FROM table1 WHERE ..
   UNION
     SELECT * FROM table2 WHERE ..
   UNION
     SELECT * FROM table3 WHERE ..
   UNION
     SELECT * FROM table4 WHERE ..
)
SORT BY col1, col2, col3     -- idem sortWithinPartition
```

Example of « latest value » cristalisation analytical query « over(partition by) »

```
INSERT OVERWRITE
        lake_team_domain.table
SELECT
   col1,col2,…. colN    -- idem * EXCEPT rank   (cf issue SPARK-33164)
FROM (
   SELECT *,
      RANK() OVER (PARTITION BY id ORDER BY update_time DESC) as rank
   FROM lake_team_domain.event_table
)
WHERE rank=1
SORT BY col1, col2, col3     -- idem sortWithinPartition
```

Example of a « Small » Data Processing resource needed = RAM



Extract all the Trades on perimeter « X »
Since <= 3 Years

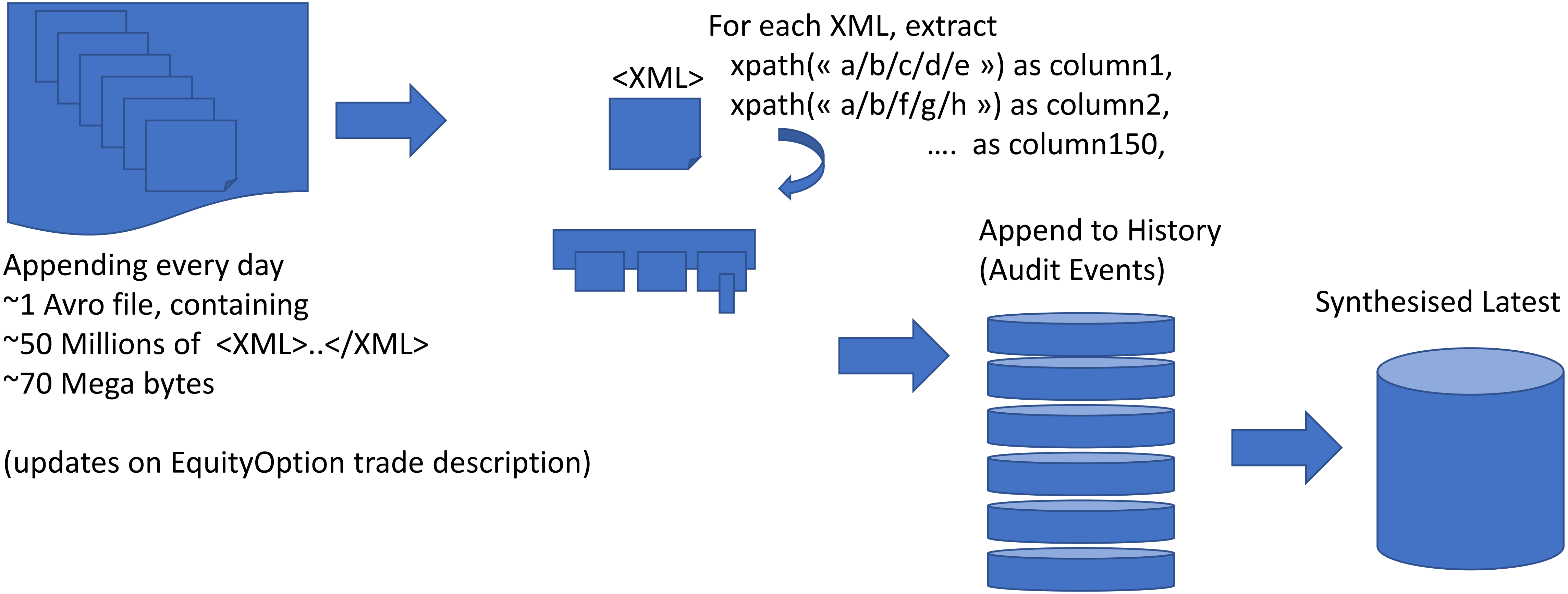
Enrich « JOIN »
With other tables

Basic Compute
SQL
Select .. where ..

Save Result
Export to OLAP Cube

Processing on Azure
using cluster 10 x D14 (100 000 euros/year)
ONLY 600 Giga of RAM data …
NOT Enough => Swapping => take 4 Hours

Example of a Slow « Un-parallelized » Processing resource needed = CPU x Time x ..



Appending every day
~1 Avro file, containing
~50 Millions of <XML>..</XML>
~70 Mega bytes

(updates on EquityOption trade description)

For each XML, extract
xpath(« a/b/c/d/e ») as column1,
xpath(« a/b/f/g/h ») as column2,
…. as column150,

Append to History
(Audit Events)

Synthesised Latest View

A Slow Processing ?

Legacy « On-Prem »

Daily Batch takes >= 15 Hours
But is mostly SINGLE-THREADED !!!

Some Parallelized intermediate parts take
>= 200 yarn containers … 10% of Cluster
… Result = ULTRA FAST !!

Backported code « as-is » to Azure HDInsight
Used Cluster of 10 x D14 cluster (40 000 euros/month)

Run NEVER Finished
… stopped in « echo count lines before insert $(…) »

Fully re-implemented in Spark + Java
instead of Hive SQL + UDF xpath
+ Optimized …

Now run in 40 minutes on 2 x D13 cluster

Outline

- What is BigData ?
  - Order of Magnitudes for « Big »
  - History
  - Evolution of Softwares, Spark
- Description of a Datalake
  - Content, data feeding
  - How it is organized
  - Who uses it
- Example of Data-Processing
  - RAW to LAKE, Reports
- Change Storage-Compute, Evolution to Cloud

Appendix:
Cloud

… More Detailed on

« From OnPrem to Azure Cloud »

see Document « -cloud.pdf »

Appendix: Cloud

Fundamental Resources TradeOffs

Network (Bandwith, Latency)

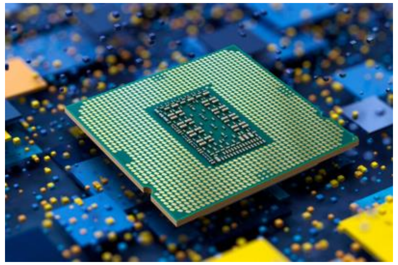
Horyzontal Scale

✕

Development price / Run cost

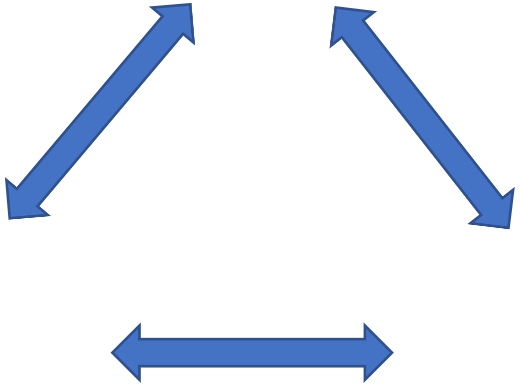
€€€€€€$$$$$

CPU (HOT, Watts)



Cold Storage (SLOW)

Storage (~Small)



RAM (FAST .. Expensive)

+ + + + + +

▬ ▬ ▬ ▬ ▬ ▬

Appendix: Cloud

Performance Changes -> Architecture Changes

Historically:
SLOW Disks
Disk Collocated with RAM+CPU

MapReduce « send program » collocated to data

Now:
FAST(ER) SSD Disks + Networks

Can « send» data to distributed programs

Storage and Compute can be SPLIT … OK !

Higher Density for Huge Storage, lower prices

48 * disks in 2U blade

1 Peta SSD … 200 000 €

Cloud Provider : AWS, Azure, Google
offer Storage solutions (S3, Azure Storage, ..)

Azure Storage Amazon S3

+ Managed Services
+ Serverless
+ Kubernetes

Appendix:
Cloud

Migration to Cloud
Azure - AWS - Google

Goals:

- Elasticity of Storage
  (No more fear of HDFS FileSystem Full)
- Elasticity of Compute
  (adapt CPU to workload… Pay only what you use)

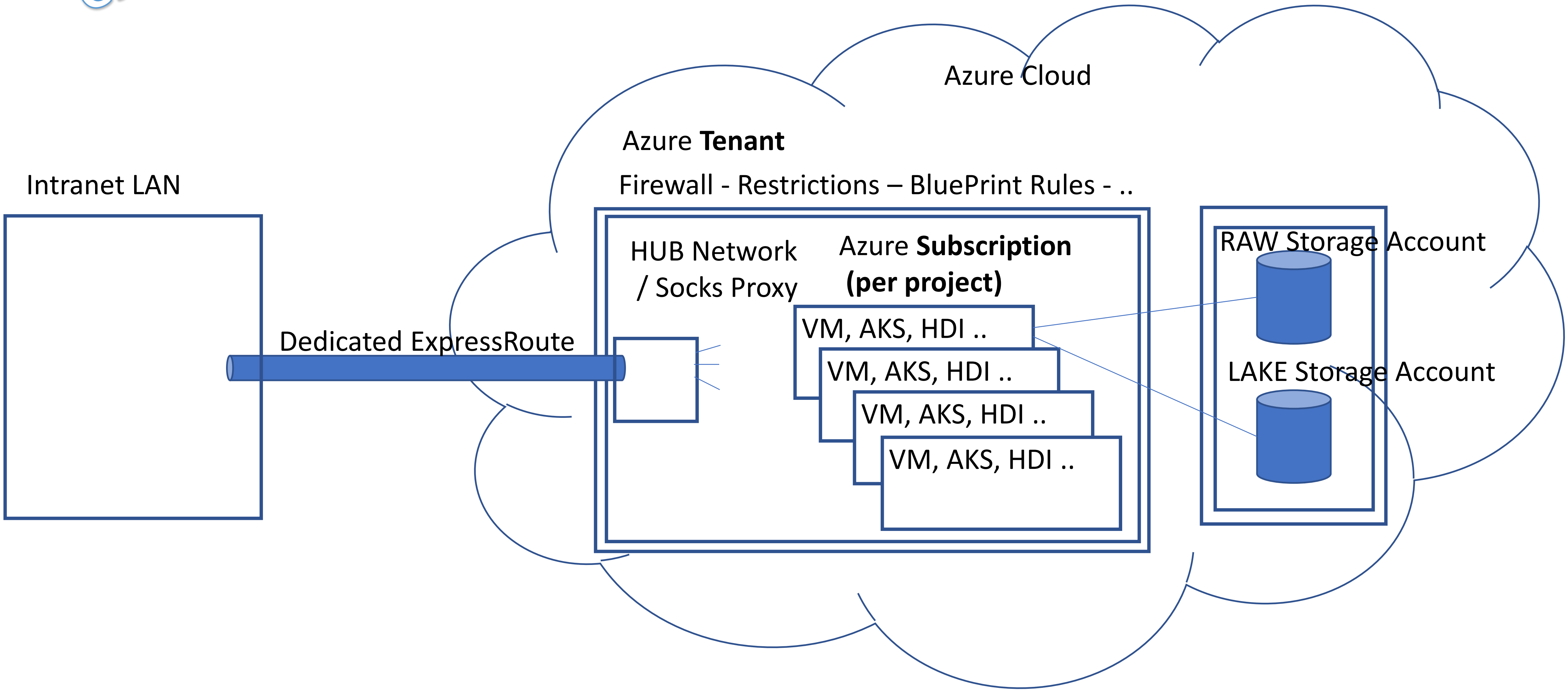
- Clear visibility of cost per Projects / internal refacturation
- By-Pass internal IT department

- Easier (?) Self-Service API for Provisionning
- No More Multi-tenant (no risk of 1 project crashing/consuming whole cluster)

Since 2020 SGCIB is moving its Datalake to Azure

Appendix: Cloud

Architecture of Datalake on Clouds



Appendix:
Cloud

BigData Engineering from OnPrem to Azure

Development of custom Yarn/Oozie/Ranger tools	Lot of Network & Security Development of Provisionning « sudo » tools
1 Huge cluster, used as Multi-tenant (several users)	NO more multi-tenant system… but LOT of small clusters
Used at 100%	Hundred of clusters, each used at 5% !!! COST COST COST Necessity to Optimize Performance (batches too long… 100x slower / too expensive)
Almost Disk-Full Many report on disk usage / data purges	No more Disk worry … data is growing (no more purges?)
Clear view of all Running workloads (1 Ambari screen)	NO admin central view NO Orchestration of workloads

Appendix: Cloud

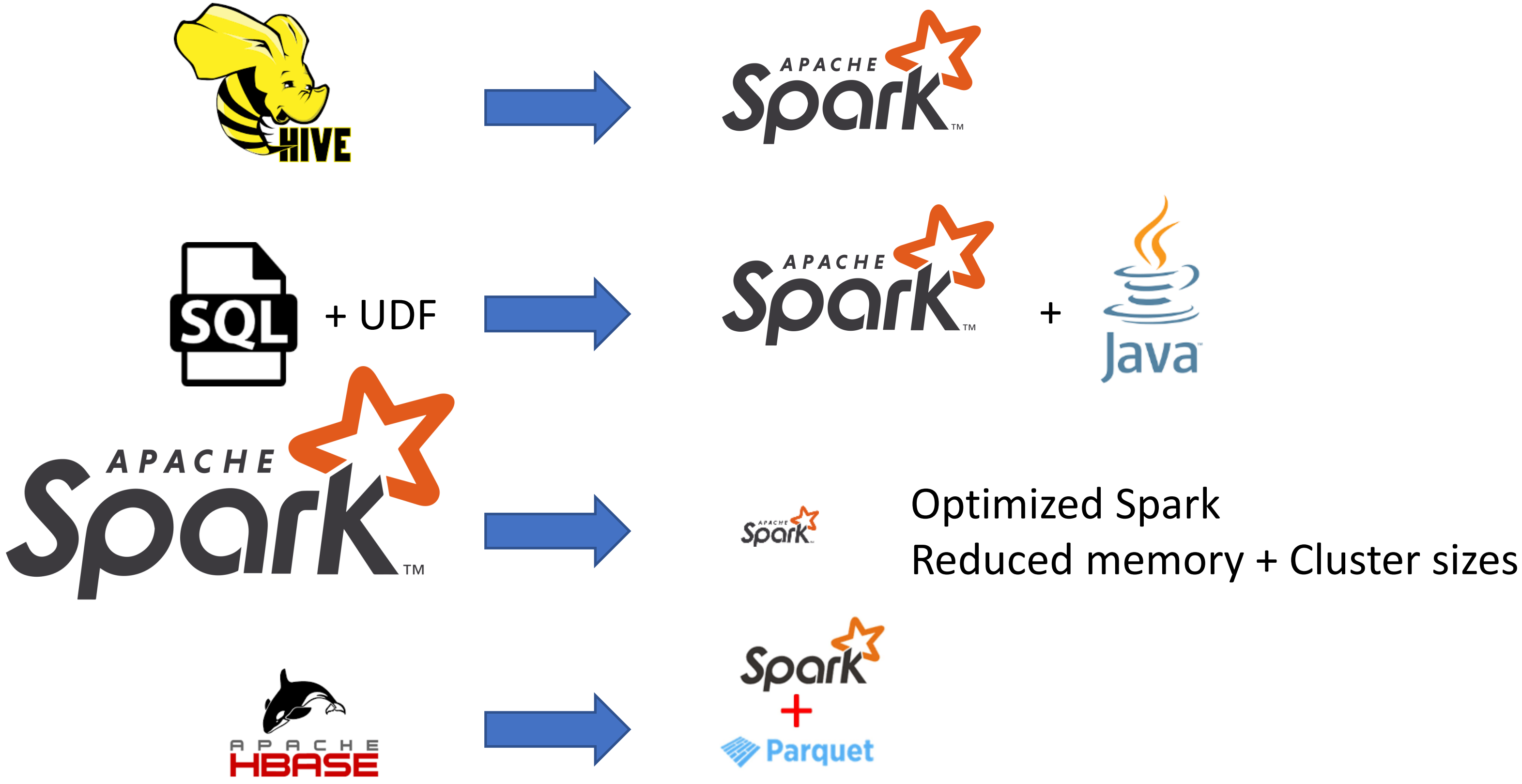
What About Spark in Azure Migration?

COST COST COST

Necessity to Optimize Performance

(batches too long… 100x slower / too expensive)

=> Many projects have
migrated + optimized



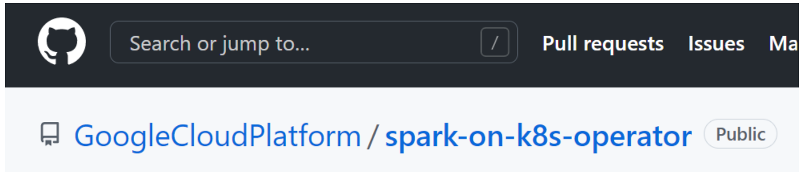
Optimized Spark
Reduced memory + Cluster sizes

Appendix: Cloud

Managed Spark ?
Goals: Autoscaling / « serverless »

Managed by Azure : Azure Synapse Analytics = APACHE Spark

Managed via Kubernetes



Search or jump to... / Pull requests Issues Ma

GoogleCloudPlatform / spark-on-k8s-operator Public

= kubernetes APACHE Spark

databricks

databricks = APACHE Spark

AWS : EMR

GCP : BigQuery

Questions?

Take Away

What is BigData ?   Horyzontal Scaling

compute: cluster with Tera of RAM used by Spark apps
storage: Petas of Files, in parquet

What is Spark ?

Simple unified Sql/Java engine
for distributed compute (Yarn/Kube)
distributed storage (HDFS/cloud)

What is Processing ?

mostly spark batches
Feeding RAW
Transforming RAW to LAKE
Consuming SQL analytics

Hadoop ecosystem is complex, Spark brings simplicity
Ecosystem is evolving (Cloud, Kubernetes)