Apache SPARK

Osman AIDEL

JI 2018

CCIN2P3

cnrs

- Introduction
- Qu’est ce que SPARK ?
- Cas d’utilisation SPARK

Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

Introduction

Besoin

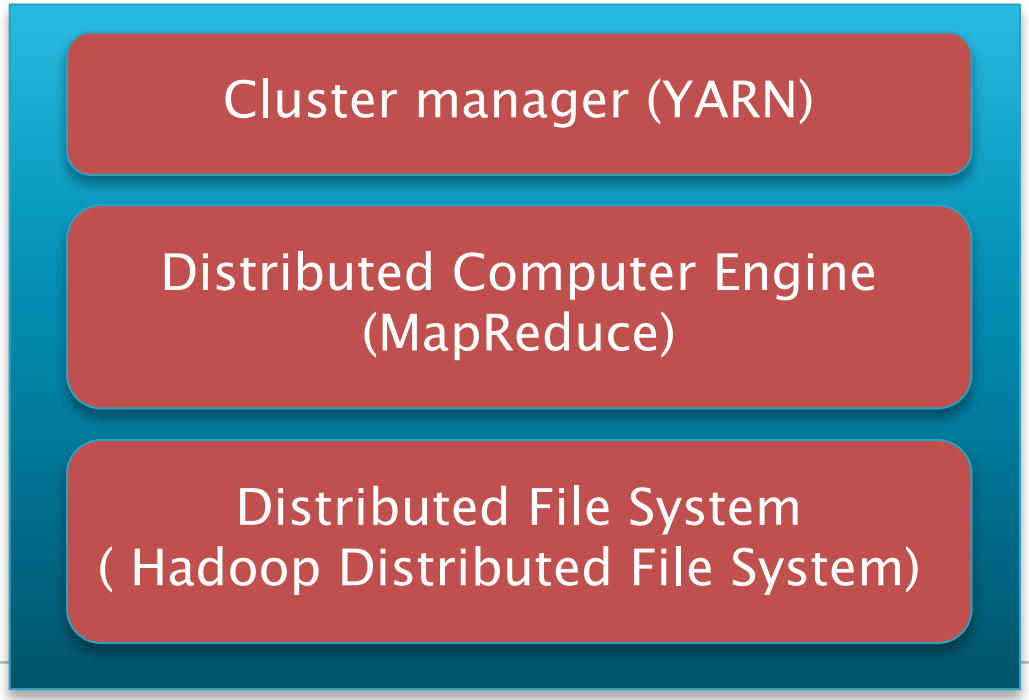
- Dans un contexte technologique ou les données sont très faciles à produire, l'analyse quant à elle devient de plus en plus complexe.
- Les premières solutions logicielles à grosse volumétrie sont commerciales et se basent sur un modèle MPP (Massively parallel processing) ou shared nothing.
  - Exemple : Teradata (1983) , greenplum (2003) …
- Solutions limitées aux grandes compagnies en raison du coût onéreux.
- Certaines compagnies privilégient une approche verticale
- En 2003/2004, google publie deux articles
  - MapReduce: Simplified Data Processing on Large Clusters
  - **The Google File System**
- Naissance de Hadoop en 2009 (Doug Cutting / Mike Cafarella)

Hadoop

- Hadoop est conçu sur plusieurs idées :
  - Développé en JAVA pour la portabilité
  - Traitement des données basé sur le paradigme Map/Reduce
  - L'utilisation de matériel commun pour le traitement et le stockage des données
  - Les pannes matérielles sont gérées au niveau logiciel
  - Exécuter le code applicatif au plus prêt des données
  - Séparer la logique du traitement des données de la logique de distribution du calcul

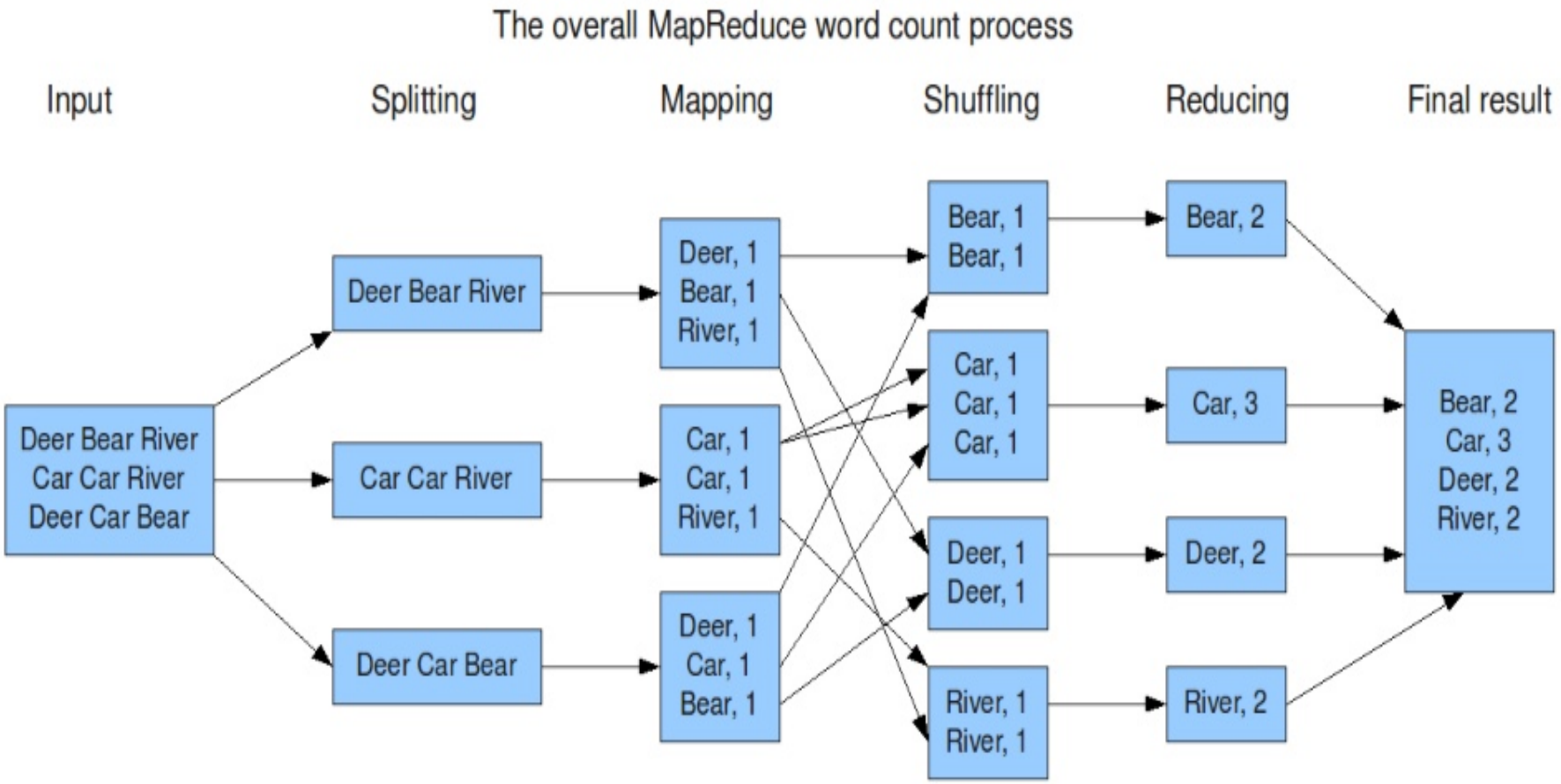
- De nombreuses organisations adoptent Hadoop en remplacement les solutions commerciales (ETL).

- Les composants clés de Hadoop 2.0



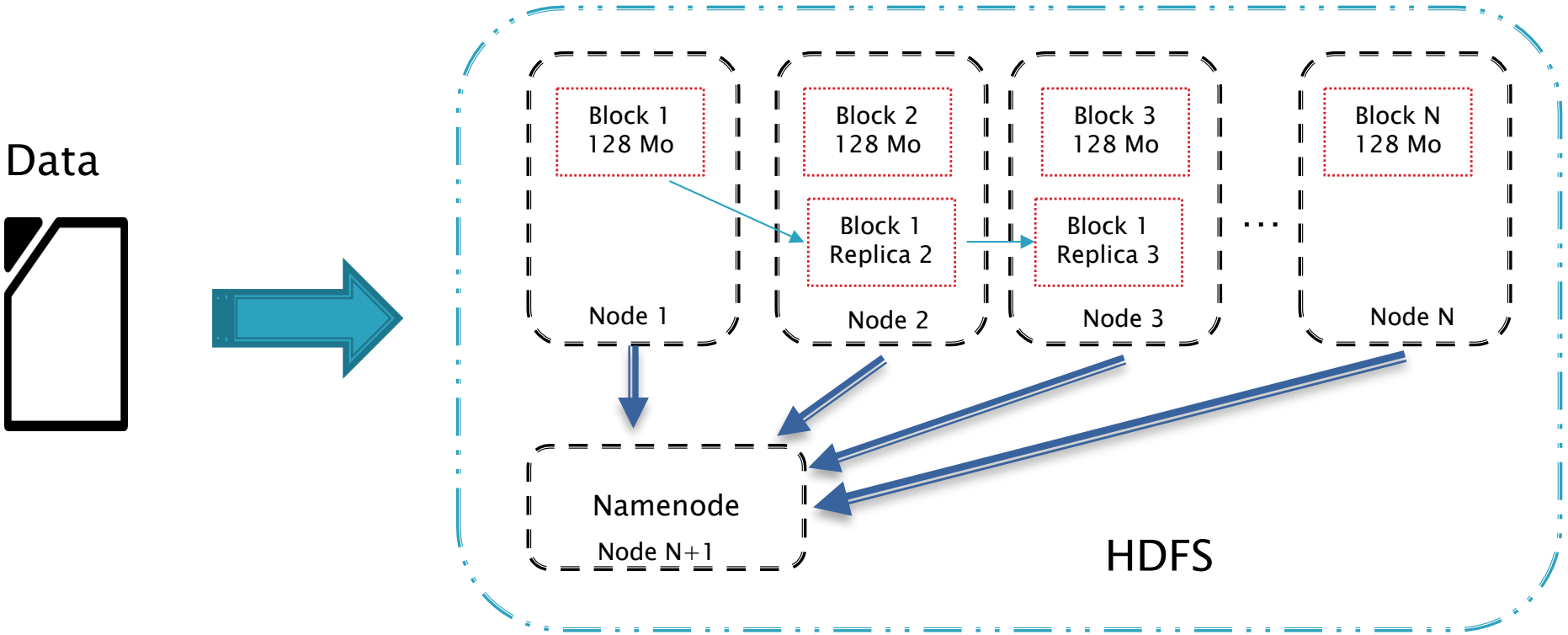
Depuis la version 2.0, Hadoop a
Adopté une architecture modulaire
Permettant d'inter-changer
les composants

Le paradigme MapReduce



2 fonctions MAP et REDUCE empruntées à la programmation fonctionnelle

Hadoop Distributed File System



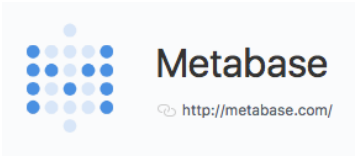
Traitement de données

Source de données
- Document
- RDBMS
- Système de fichiers
- …

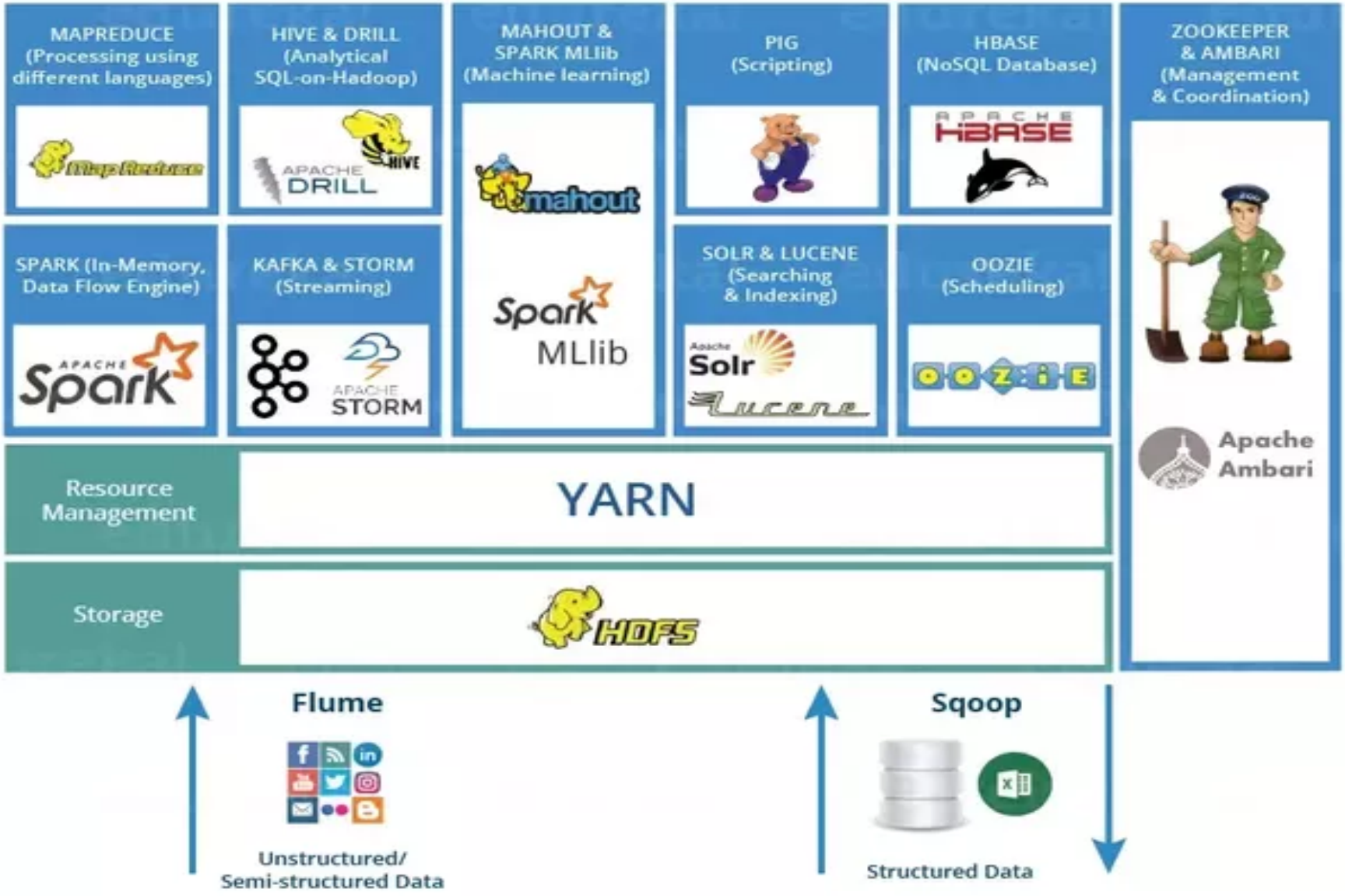
Couche d'intégration
- Connexion
- Collecte
- Correction
- Transformation

Couche d'analyse
- Exploration
- Machine Learning
- Graphe
- Streaming

Visualisation
- Report
- Graphique
- Time series



Hadoop


MAPREDUCE
(Processing using different languages)
HIVE & DRILL
(Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib
(Machine learning)
PIG
(Scripting)
HBASE
(NoSQL Database)
ZOOKEEPER & AMBARI
(Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM
(Streaming)
SOLR & LUCENE
(Searching & Indexing)
OOZIE
(Scheduling)
Resource Management
YARN
Storage
HDFS
Apache Ambari
Flume
Unstructured/
Semi-structured Data
Sqoop
Structured Data

Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

SPARK

Qu'est ce que SPARK ?

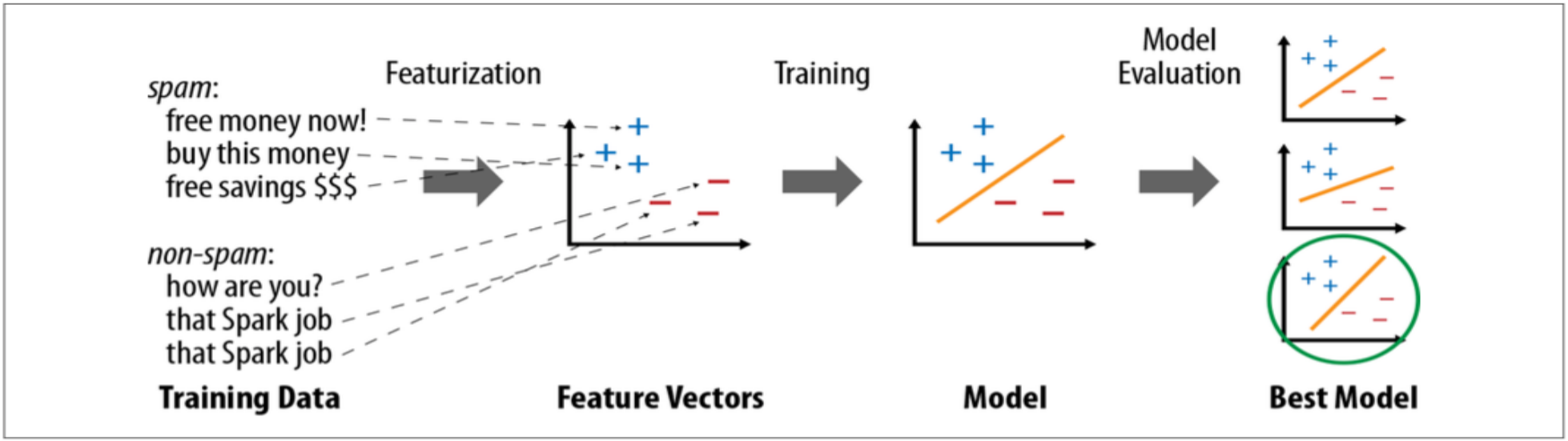
- SPARK s'inspire fortement de Hadoop
  - L'utilisation de matériel commun pour le traitement mais ne gère pas le stockage des données
  - Les pannes matérielles sont gérées au niveau logiciel
  - Exécuter le code applicatif au plus prêt des données
  - Séparer la logique du traitement des données de la logique de distribution du calcul
  - Intègre son propre cluster manager 'SPARK standalone' lequel est interchangeable avec YARN / MESOS

- Les améliorations apportées par SPARK
  - Une évolution améliorée du moteur MapReduce
    - Spark est un cluster dit in-memory
    - Un optimiseur paresseux et intelligent
    - Considéré comme le successeur de MapReduce
  - Développé en SCALA, langage basé sur une JVM.
    - 100 lignes de code JAVA peut être traduit en une dizaine de lignes en SCALA
  - Hérite des avantages de la programmation fonctionnelle
    - Le calcul est évalué comme une fonction mathématique
    - Le développement d'applications parallélisées est plus facile
  - Concept de Resilient Distributed Dataset

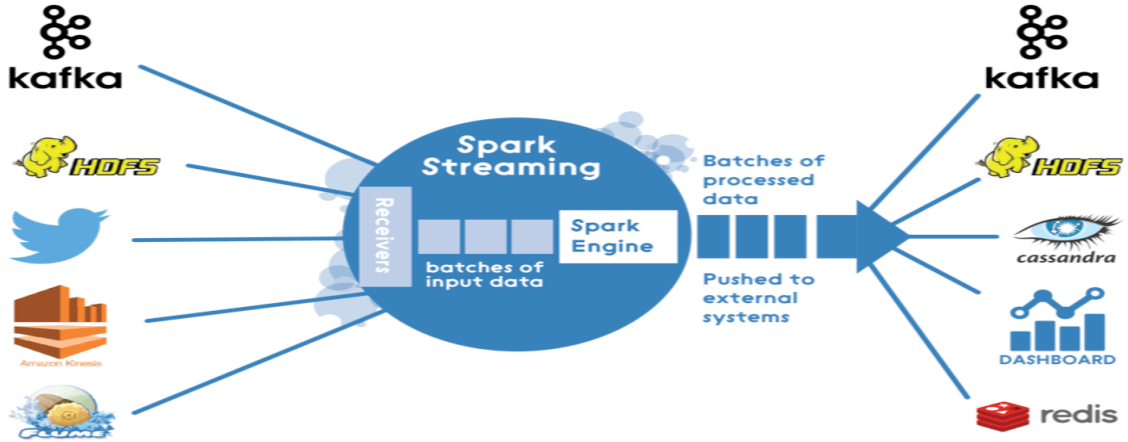
Qu’est ce que SPARK ?

- Un cluster de calcul très populaire
- Plateforme proposant une API JAVA, PYTHON, SCALA
- Fonctionnalités
  - Batch / MapReduce
  - Machine Learning
  - Streaming
  - Spark SQL
  - GraphX

- Mllib

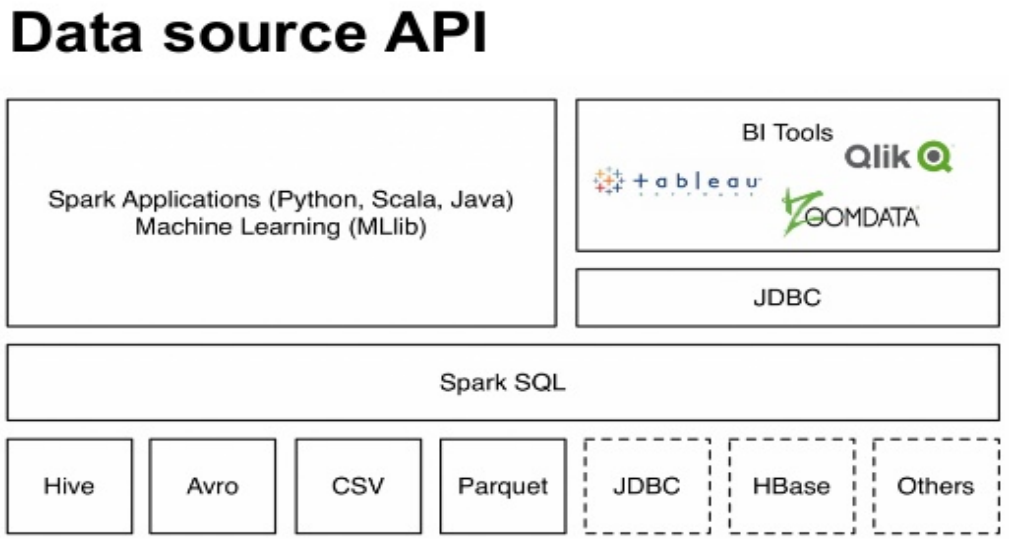


- Streaming : Fume, kafka

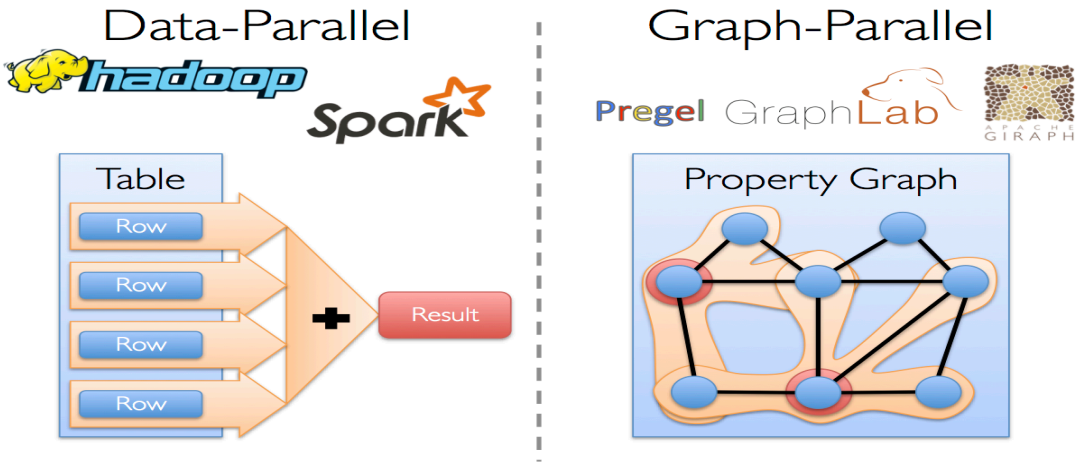


Spark features

- Spark SQL



- GraphX

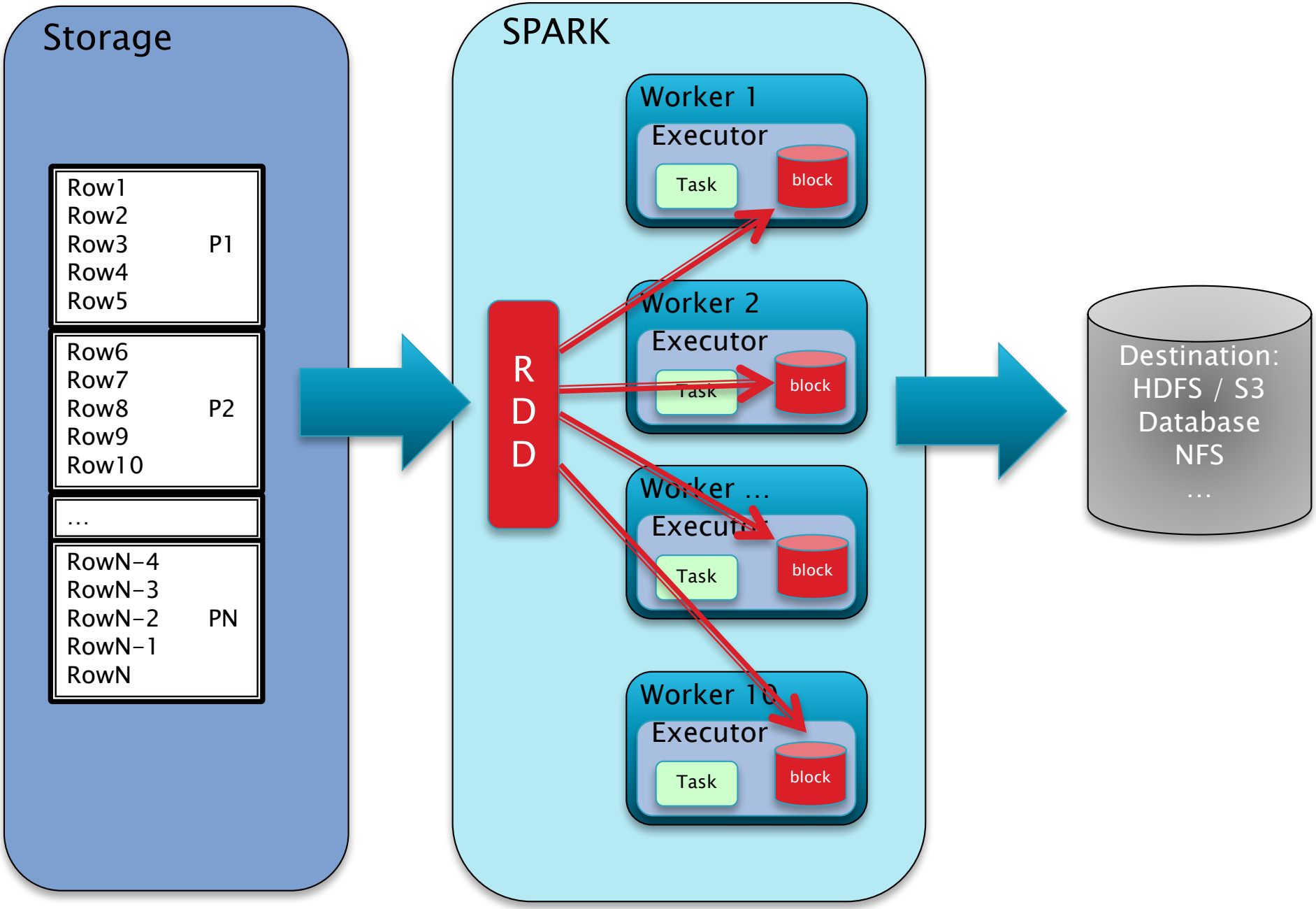


Comment cela fonctionne t'il ?

Compter le nombre de caractères dans un fichier

```
dataSrc = sc.textFile("data.txt")
lineLengths = dataSrc.map(lambda s: len(s))
totalLength = lineLengths.reduce(lambda a, b: a + b)
totalLength.collect()
```

Comment cela fonctionne t'il ?



Qui l'utilise ?

- Le CERN
  - Développement d'un plugin pour lire les fichiers ROOT
- LAL / U-PSUD
  - Service en production ouvert à la biologie et aux physiciens.
  - Développement d'un plugin pour lire les fichiers FITS
- NERSC
  - Sur CORI (Shifter)
- CDS de Strasbourg
  - Veille technologique sur le cross-matching des catalogues d'astronomie

Ce qu'il faut retenir de SPARK

- SPARK est facile d'utilisation
- Le développement de votre programme est très rapide et passe rapidement à l'echelle
- Les performances d'exécution ne peuvent pas rivaliser avec un programme optimisé
- De nombreux plugins sont disponibles pour faciliter le chargement et le traitement des données (Cassandra, MongoDB, JDBC, CSV, AVRO, PARQUET…)

Spark projects

- IBM Spectrum Conductor with Spark
- Apache Mesos : Cluster management system that supports running Spark

- Spark Job Server : interface REST for submitting jobs
- EclairJS : interface web

- Alluxio (née Tachyon) - Memory speed virtual distributed storage system that supports running Spark

- Spark Cassandra Connector - Easily load your Cassandra data into Spark and Spark SQL; from Datastax
- FiloDB - a Spark integrated analytical/columnar database, with in-memory option capable of sub-second concurrent queries
- ElasticSearch - Spark SQL Integration
- SnappyData fuses Apache Spark with an in-memory database to deliver a data engine capable of processing streams, transactions and interactive analytics in a single cluster.

- GeoSpark - Geospatial RDDs and joins

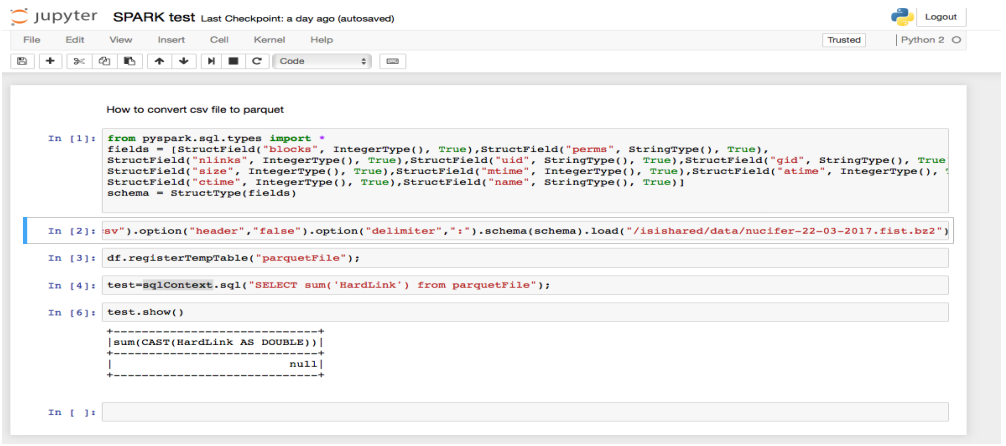
- DF from Ayasdi - a Pandas-like data frame implementation for Spark
- Oryx - Lambda architecture on Apache Spark, Apache Kafka for real-time large scale machine learning
- ADAM - A framework and CLI for loading, transforming, and analyzing genomic data using Apache Spark

- Beaucoup de projets notamment dans des frameworks de Deep / Machine learning basé sur SPARK

Interface utilisateur

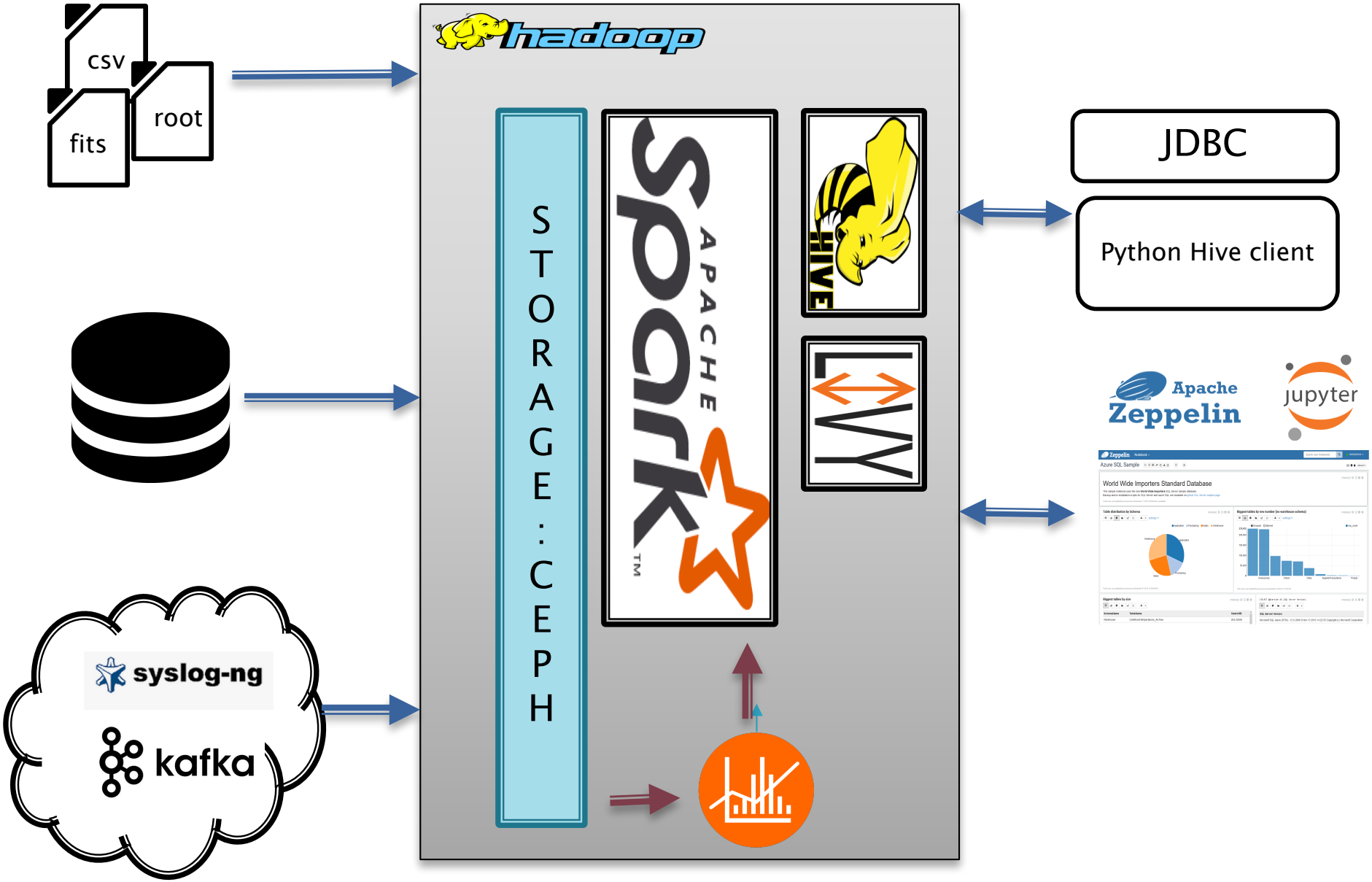
- Spark client
  - Spark-submit (mode batch)
  - Pyspark /spark-shell (interpreter)

- Jupyter / Zeppelin



- Spark Job Server / Livy : interface REST for submitting jobs

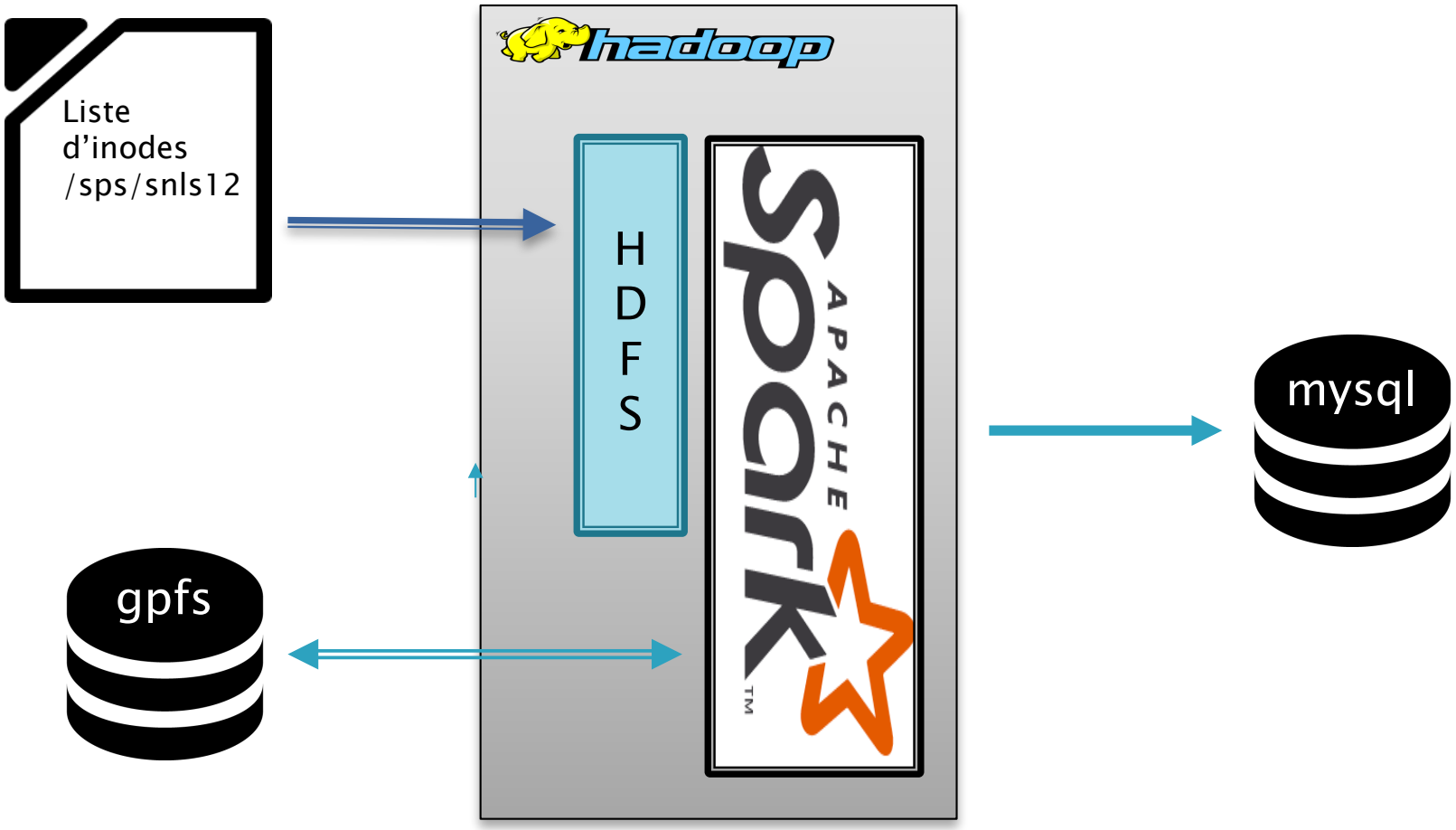
SPARK au CC


csv
root
fits
hadoop
STORAGE : CEPH
APACHE
Spark
TM
HIVE
JDBC
Python Hive client
Apache
Zeppelin
jupyter
syslog-ng
kafka

POC SPARK

Un catalogue central des métadonnées FITS pour LSST

Catalogue FITS



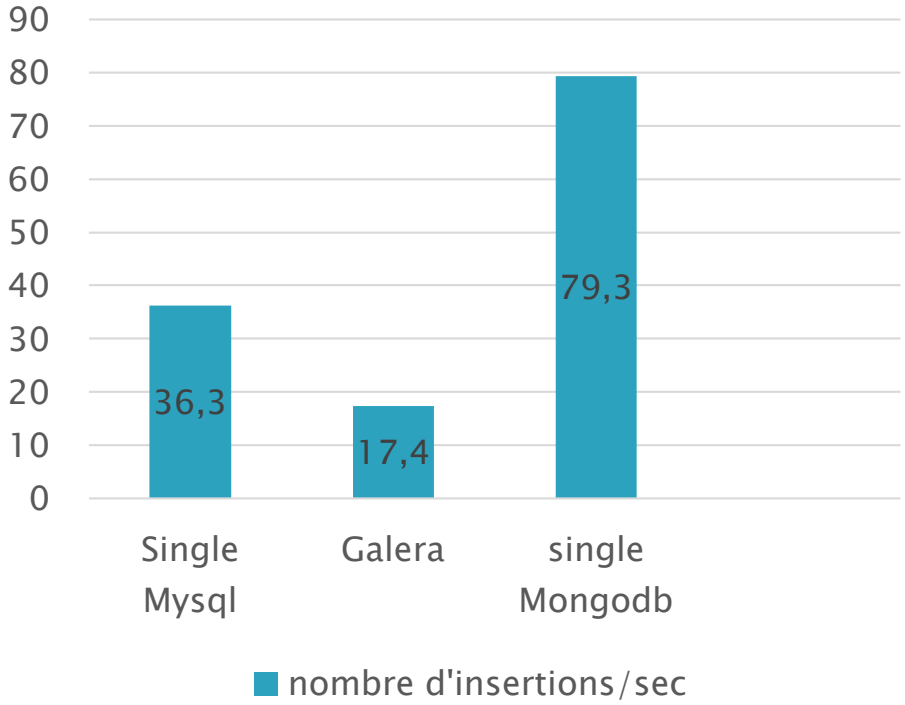
Catalogue central des métadonnées FITS

- 9 140 782 de fichiers FITS

- Single Mysql server
  - 1 machines 16Gb of RAM and 8 cores
  - Storage engine : MyIsam
  - No compression

- Galera cluster
  - 3 machines 32Gb of RAM and 16 cores per machine
  - Storage engine : Innodb
  - 400Gb
  - No compression

- Mongodb
  - 1 cloud VM : 4Gb of RAM and 4 cores
  - Storage engine WiredTiger (new from 3.0 release)
  - 46Gb
  - Compression : snappy



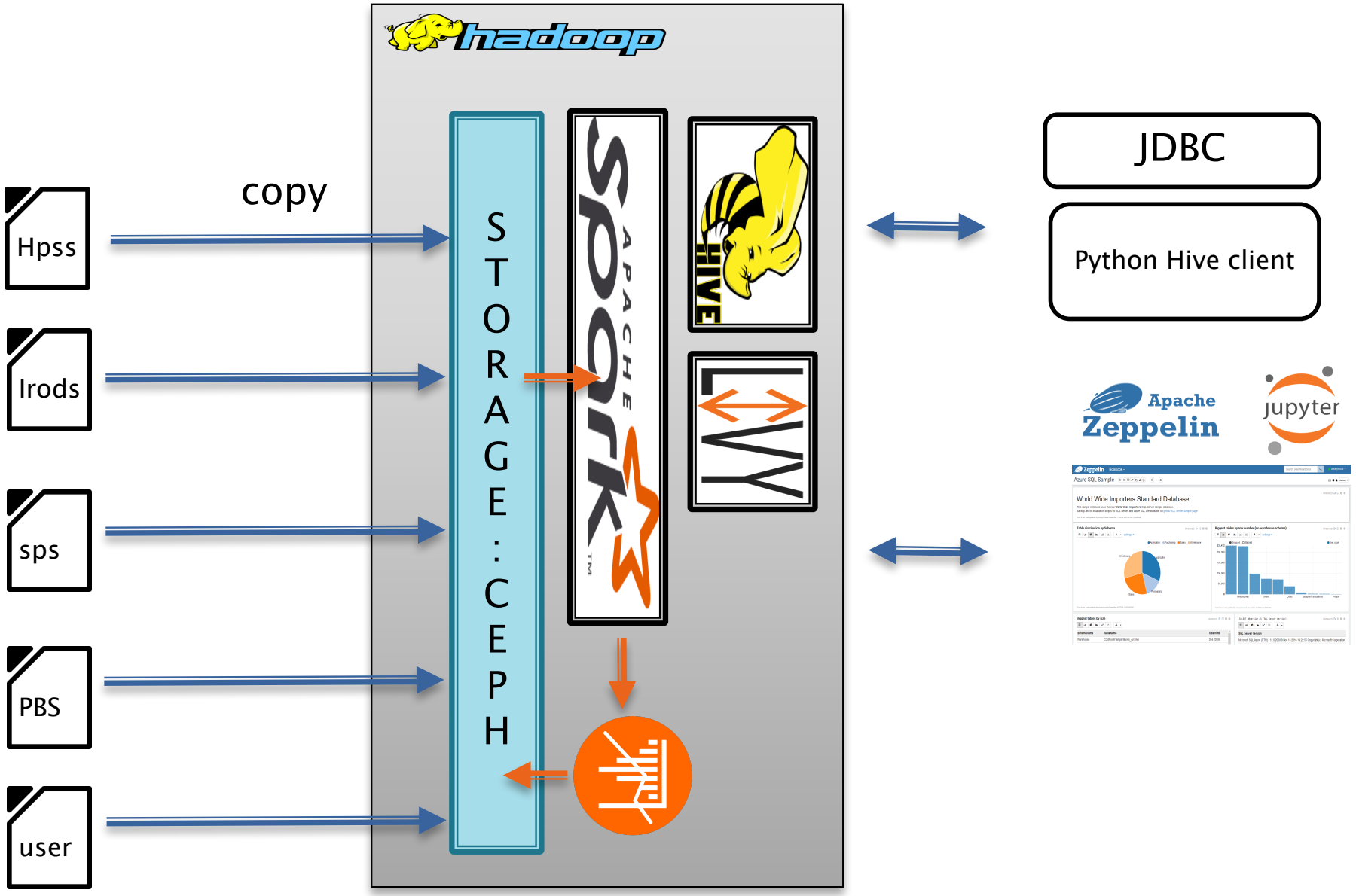
POC SPARK

Statistiques de stockage

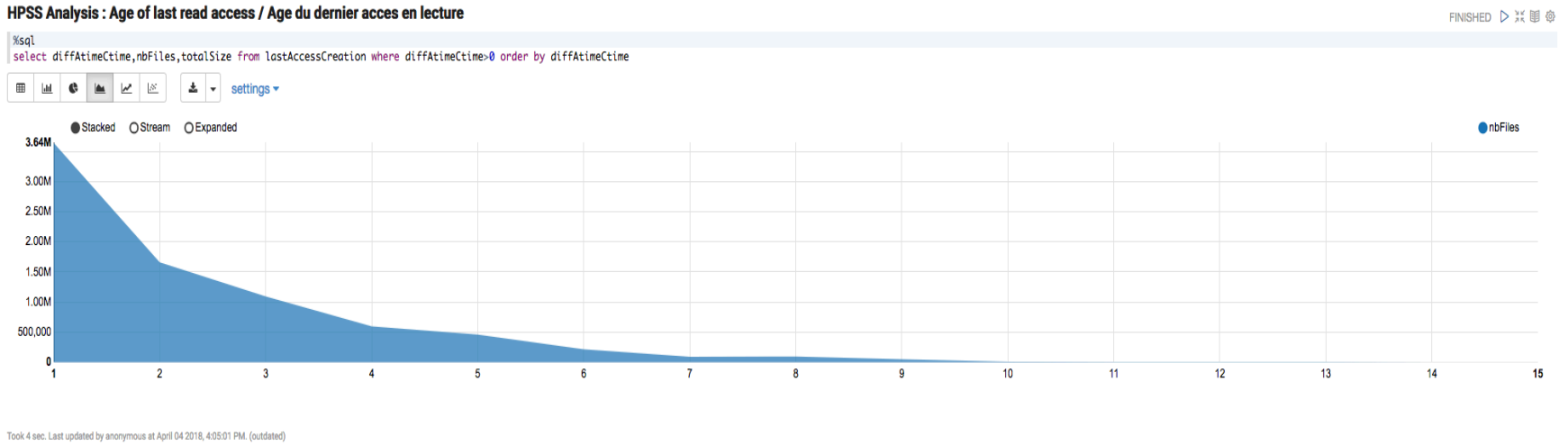
Statistique pour le stockage

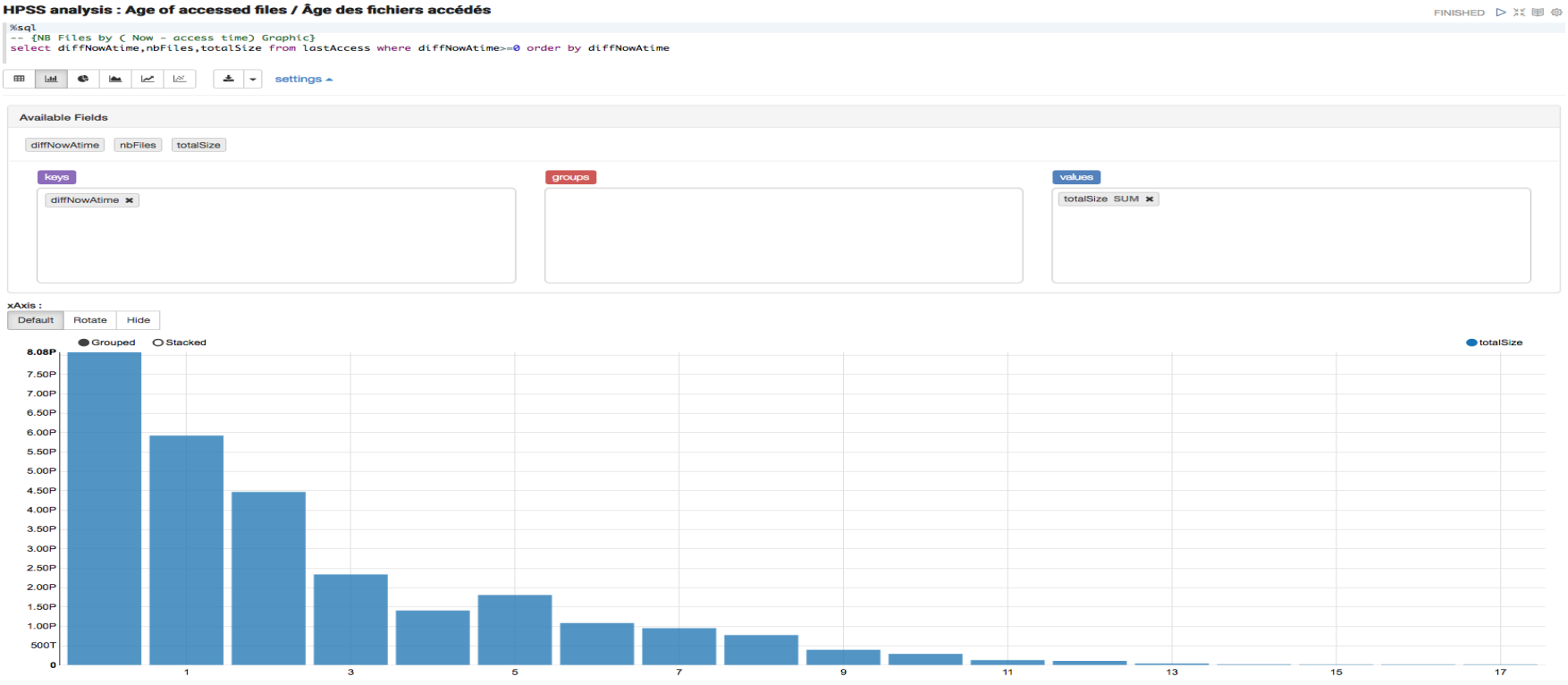
- Objectif
  - Rendre consultable les métadonnées du stockage du CC :
    - Centraliser les métadonnées
    - Traitement des données en mode batch ou en stream
  - Système de stockage:
    - SPS, PBS, IRODS, HPSS, xROOTD …
    - Référentiel d'identité
  - Vue support
    - Les fichiers non accédé depuis plus d'un an
    - Identifier les fichiers d'un utilisateur quelque soit le groupe
    - Identifier le top 10 des répertoires par groupe / utilisateur
    - …
  - Vue expert
    - Taille totale /moyenne des fichiers par groupe / groupe-utilisateur
    - Nombre de fichiers / de répertoires / de liens par groupe / groupe-utilisateur
    - Les fichiers les moins accédés
    - …
  - Vue responsable
    - Etat des différents système de stockage d'un point de vue global / groupe / utilisateur
    - Respect des DMP : nombre de réplicas de fichiers …

SPARK au CC


hadoop
copy
Hpss
Irods
sps
PBS
user
STORAGE:CEPH
APACHE
Spark
HIVE
JDBC
Python Hive client
Apache
Zeppelin
jupyter

Exemple de visualisation





Exemple de visualisation

gid	uid	nbfiles	sum(size)	max(size)	min(size)	avg(size)	min(atime)	max(atime)	min(mtime)	max(mtime)	nblinks	nbdirs	nbHardLinks
124	2482	928	1.50252583E8	7683467.0	0.0	161910.1109913793	1473762577	1473762577	956041178	974133185	41	55	0
246	41713	24	1.115839081E9	1.093149253E9	634.0	4.6493295041666664E7	1354645342	1511531084	1354645098	1362516979	0	13	0
269	40647	416195	5.47885353524E11	1.444582595E10	0.0	1316415.0302718678	1133946789	1516827361	829617606	1463579877	6786	46653	727
269	42604	1669	1.81440135E8	1.1745607E7	2.0	108711.88436189335	1444051156	1507620099	1310039491	1507619924	0	68	0
138	2513	5143	1.41621786E8	2.4745204E7	0.0	27536.804588761424	1456410006	1456410006	1023965601	1028107557	0	14	0
1100	42564	20356	1.6777942158E10	9.490032E8	0.0	824225.8871094518	1438874723	1516836085	1343060073	1516836067	9	1791	0
144	41340	9	1.019507339E9	1.15746362E8	1.11929908E8	1.1327859322222222E8	1404141503	1404141522	1315988668	1315989143	0	1	0
1023	40566	3161784	5.1410851291E11	7.12098148E8	158.0	162600.76997985947	1236507805	1366627337	1236340082	1336658463	0	10588	0
1013	40216	3916677	4.95778327622E11	2.724815443E9	0.0	126581.36671009634	700539125	1516836017	700539125	1516718642	695	74627	0
102	42251	2	3.7580965793E10	3.758096384E10	1953.0	1.87904828965E10	1505202733	1516196026	1504177498	1516196027	0	1	0
1016	41437	61115	1.19535797631E11	1.734781112E9	0.0	1955915.8574981592	1327326112	1335440143	1327326112	1335440143	0	5421	0
209	41100	263375	2.217679137024E12	7.9886062677E10	0.0	8420234.027618416	1415374452	1516835109	1015511691	1516835103	101	7995	0
133	2321	1757	1.045392339E9	1.3839464E7	0.0	594987.1024473534	1489064136	1489065630	791513040	1060229764	4	40	0

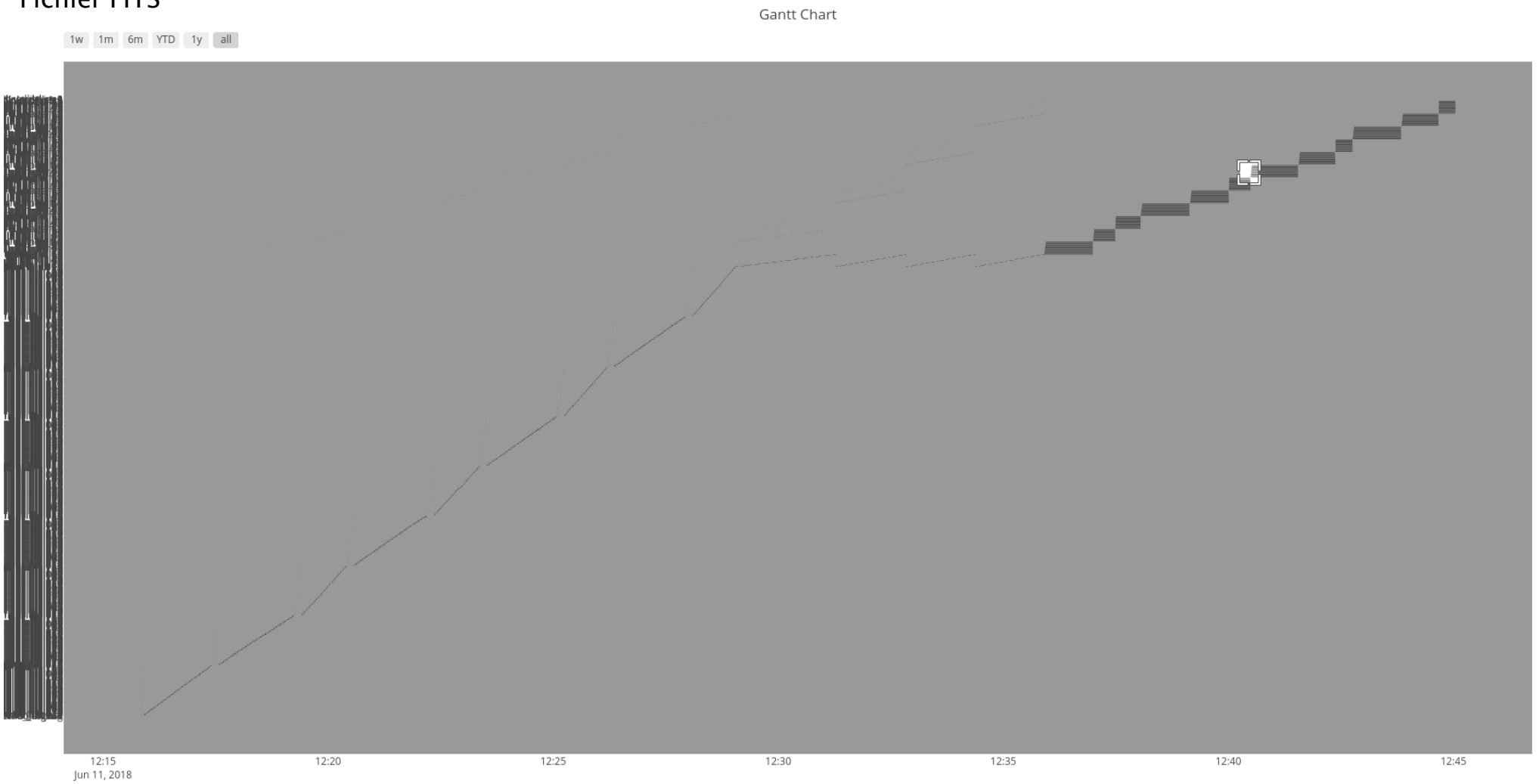
uid	directory	nbfiles	Maxdepth	MinDepth	sumSize
1780	/xrootd/store/SPskims	637221	10	8	2.75335132035287E14
1780	/xrootd/store/cfg	19	7	7	2.06213266E8
1780	/xrootd/store/PR	82940	10	10	5.555392462802E13
1780	/hpss/in2p3.fr/home	39	8	7	5.840119752E10
1780	/xrootd/store/SP	427076	9	6	2.72561375827162E14
1780	/xrootd/store/testNgop.root	1	4	4	1.662047417E9
1780	/xrootd/store/cdb	8410	10	10	7.90985683316E11
1780	/xrootd/store/PRskims	321955	9	8	1.35911759077134E14

POC SPARK

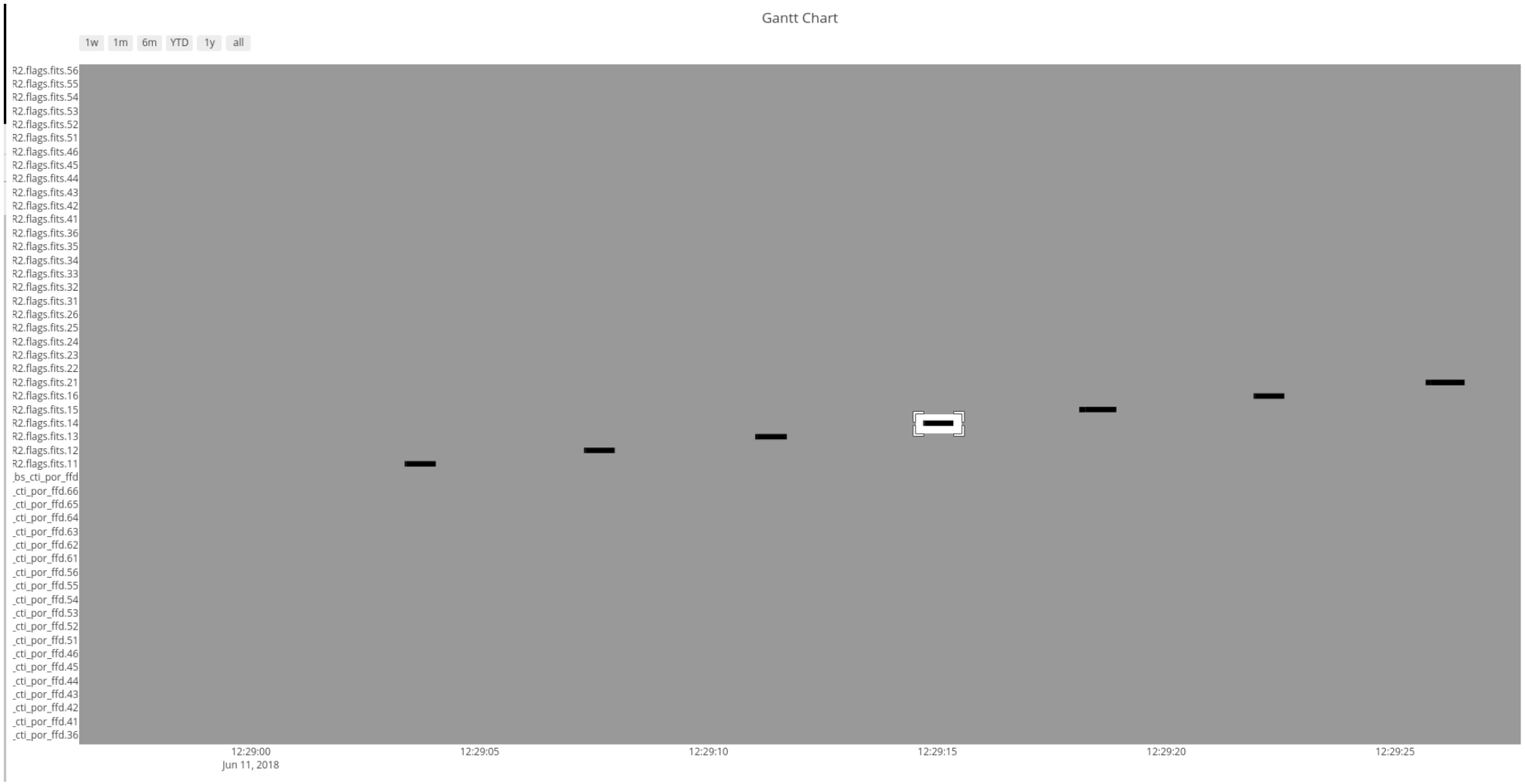
Euclid : analyse de jobs

Euclid : analyse de jobs

Fichier FITS



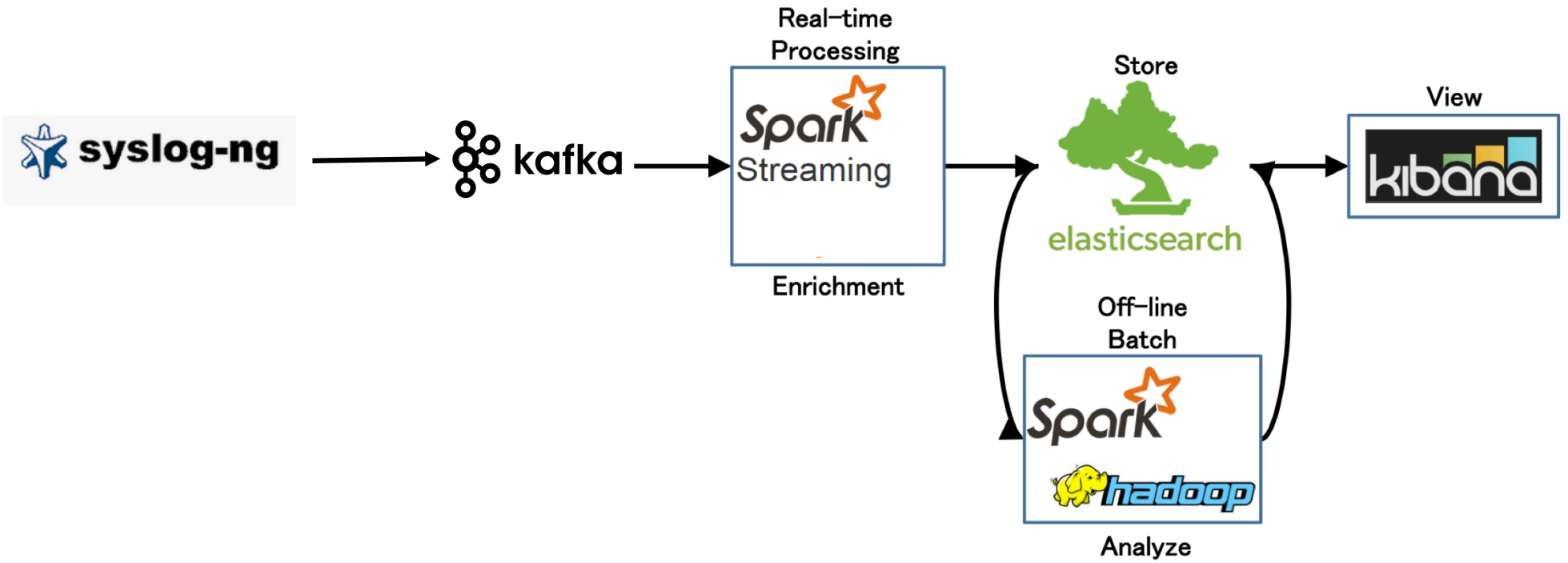
Euclid : analyse de jobs



POC SPARK

Analyse de comportements frauduleux

Analyse de comportements frauduleux



Conclusion

- Séparer la logique du traitement des données de la logique de distribution du calcul
- Rendre le traitement des données « scalable » quelque soit la volumétrie
  - Gros volume de données
  - Historique des données
- Faciliter l'exploitation des données
  - Interface utilisateur simple (zeppelin / Jupyter)
  - API SQL / Scala / Python
  - Pour des soucis de performance, Python n'est pas recommandé
- SPARK fait partie intégrant de l'écosystème HADOOP, lequel privilégie le système de fichiers de stockage HDFS ou S3.
- Dans un environnement virtuel, le stockage peut être distant et l'usage / la pertinence d'HDFS n'a pas trop de sens:
  - Distribution du calcul suivant l'emplacement des données
  - S3 semble une bonne alternative
- La conversion des données en un format parquet ou avro est d'autant plus intéressante que le nombre de requêtes est importante.

Conclusion

- Attention, la multitude de logiciels fournit dans l'éco-système hadoop est très variée et les versions évoluent très vite.
- Si vous êtes intéressé par la DATA SCIENCE, Horton Data Plateform est un bon point de départ.
- Nous étudions la mise en place d'une plateforme SPARK au CC.
  - Comment mettre à disposition ce service
  - Comment gérer l'allocation des ressources en mode multi-utilisateurs
- Si vous avez un cas d'utilisation dans lequel SPARK peut apporter une plus value, vous êtes les bienvenus.