SPARK@VirtualData

Adrien.Ramparison@lal.in2p3.fr

Workshop France Grilles Cloud. 29 juin 2018

PLAN

Équipement de Recherche Mutualisé

- Programmes ERM U-PSUD 2015-2017, 2017-2019
- Objectif : fournir un environnement d'expérimentation réaliste des traitements dits ***Big Data*** intégré dans la plateforme cloud@VirtualData
- Plusieurs entités : LRI (informatique), LESE (Ecologie),LAL, I2BC, INSERM, SHPEM (Signalisation Hormonale . . .), UUI Lipide (Chimie), DI Cellule calcul scientifique
- $\Rightarrow$ Cluster Apache Spark

Hadoop Map Reduce

- Ensemble de données (Ex : 15 TB)
- Calcul à faire sur cet ensemble : le problème se décompose bien (embarassingly parallel)
- Une seule machine ne suffit pas
- Solution : 15 machines, données partitionnées en 15, une machine 1/15e des données, calculs partiels, agrégation du résultat final
- $\Rightarrow$ concept **Map/Reduce**
  - Map : lecture des données, production Clé/Valeur
  - Reduce : Regroupement des clés, traitement des valeurs
- **Système de répartition des calculs**

HDFS

- Système de fichiers répartis
- Non posix
- Découpage de fichiers en blocs (128MB ou 256MB, configuration globale). Peut-être redéfini par l'utilisateur (variable).
- Blocs répliqués (réplica 3, configuration globale). Peut-être redéfini par l'utilisateur (variable).
- Architecture : Namenode/Datanodes
- $\Rightarrow$ **Système de répartition des données**

Apache SPARK

Spark en quelques points :

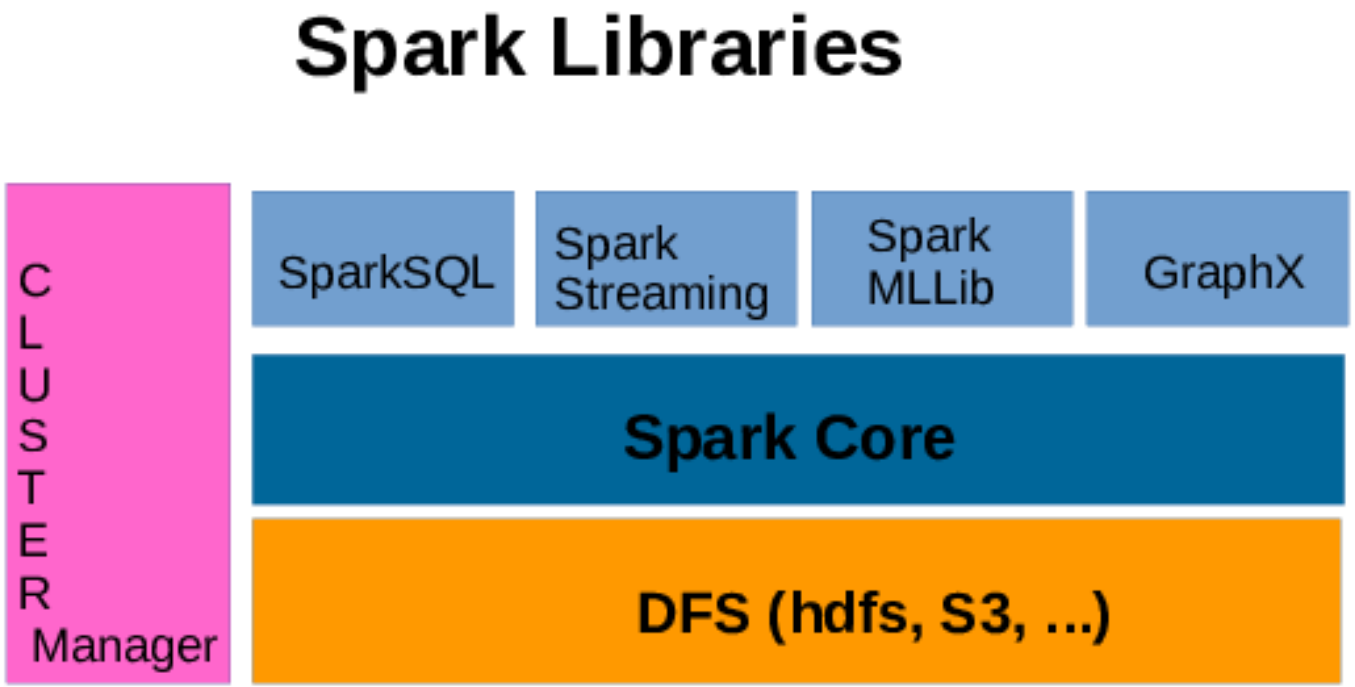
- Apache Spark : Framework pour le calcul distribué
- Similaire à Hadoop MapReduce
- Écrit principalement en scala
- Hadoop MR : écriture intermédiaire sur disque entre opérations.
  Spark : In-Memory si possible (Dataset (RDD), résultats).
- 2 opérations :
  - **Transformations** : [map(), filter(), groupByKey(), partitionBy(), . . .],
  - **Actions** : [reduce(), count(), collect(), saveAsTextFile(),. . .] .

Apache SPARK suite

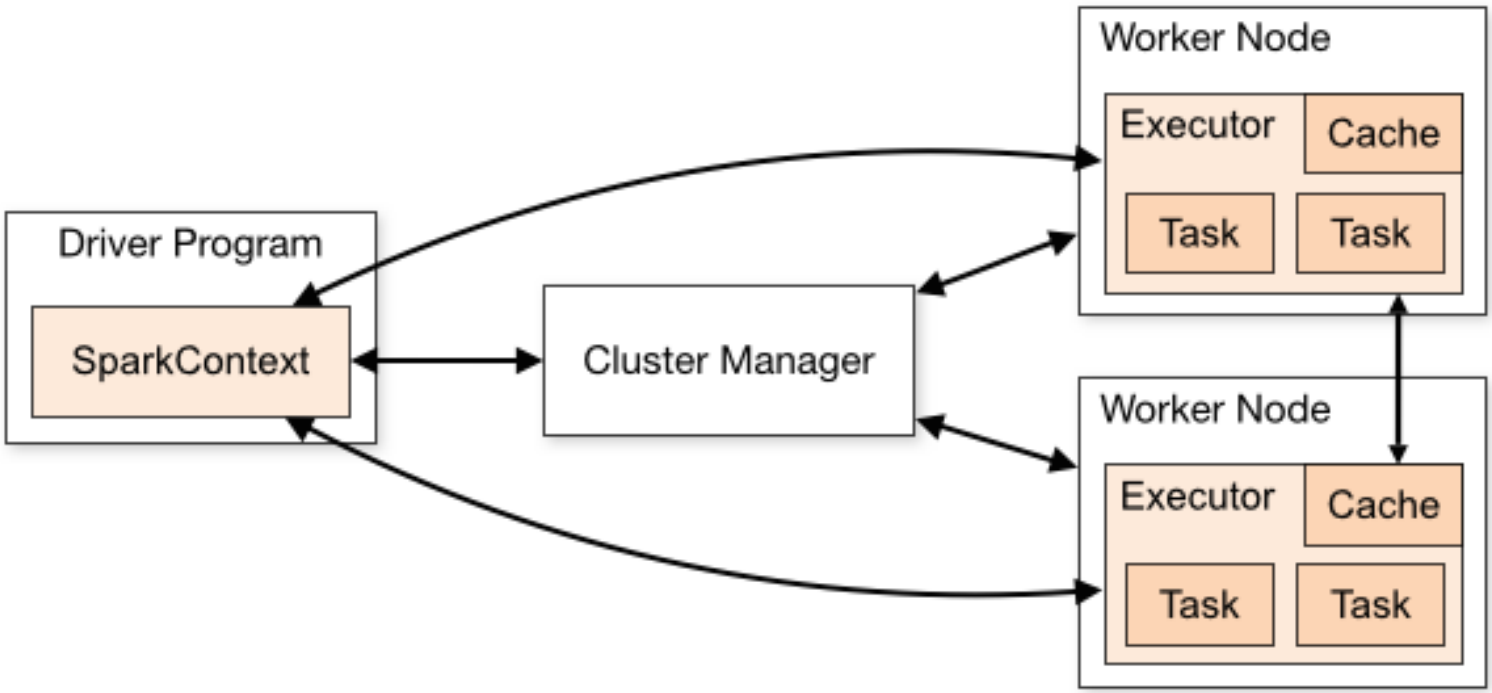
Spark en quelques points (suite) :

- Graphe Acyclique Dirigé (DAG).
- Exécution paresseuse (laziness). Exécution quand il y a action seulement.
- Le calcul se fait là où se trouvent les données (tant que c'est possible).
- Librairies natives pour ML, Streaming, R, Graph

Composants SPARK

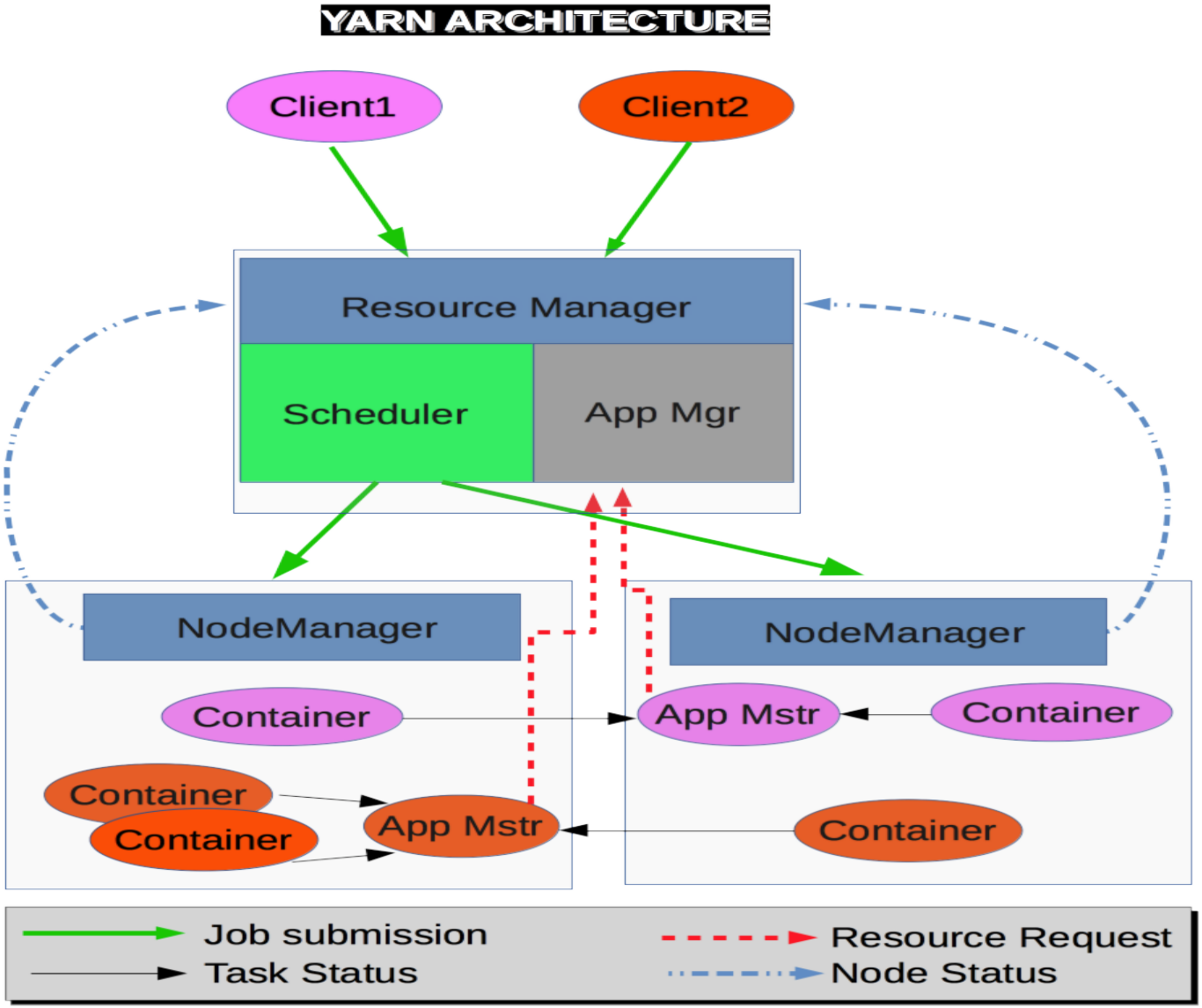


Architecture SPARK



- Cluster Manager
  - Standalone : FIFO
  - Yarn : Capacity Scheduler, Fair Scheduler
  - Mesos : Coarse-grained mode, Dynamic Sharing

Cluster Manager : YARN



INFRASTRUCTURE Spark au VirtualData

- Infrastructure à base de VMs
- Cluster de production (version stable)
  - Openstack : Domain **u-psud**, Project **Spark**, CentOs7
  - 1 Master : Gabarit 18 cores/36GB RAM **os.18**
  - 1 Yarn Cluster Manager : Gabarit 18 cores/36GB RAM **os.18**
  - 9 Slaves : Gabarit 18 cores/ 36GB RAM, volume cinder HDFS **4+2 To**
- Cluster de développement (version unstable)
  - Openstack : Domain u-psud, Project DI, CentOs7
  - 1 Master/yarn CM : Gabarit os.6
  - 3 Slaves : Gabarit os.6, volume cinder 300Mo
- Cluster de test (version testing)
  - Openstack : Domain u-psud, Project DI, CentOs7
  - 1 Master/yarn CM/Hue web UI : Gabarit os.4
  - 3 Slaves : Gabarit 4 cores/8GB RAM os.4, volume hdfs 24 GO

SERVICES

- API Standards : Scala, Python, Java
- Composants intégrés : SparkR, Spark SQL, MLLib, Spark GraphX
- Système de fichiers distribué HDFS sur ceph
- Langages : scala, java, python, R, ...
- Accès : ssh sur le **MasterNode** (nécessite clé ssh)
- Authentification U-PSUD
- Installation : scripts shell masters/workers, playbooks ansible

Soumission de jobs

UTILISATIONS

- Projet SEQOIA : Collaboration U-PSUD, AP-HP, Lateral Thoughts.
  But $\Rightarrow$ PoC Séquençage génome (medium dataset) $<$ 24h pour 6 workers.
- Formations : Spark Essentials (python), programmation fonctionnelle scala. LAL et Lateral Thoughts.
- Workflow LSST (LAL). 2 articles. Cf slides suivants.

Cluster Spark: Utilisation

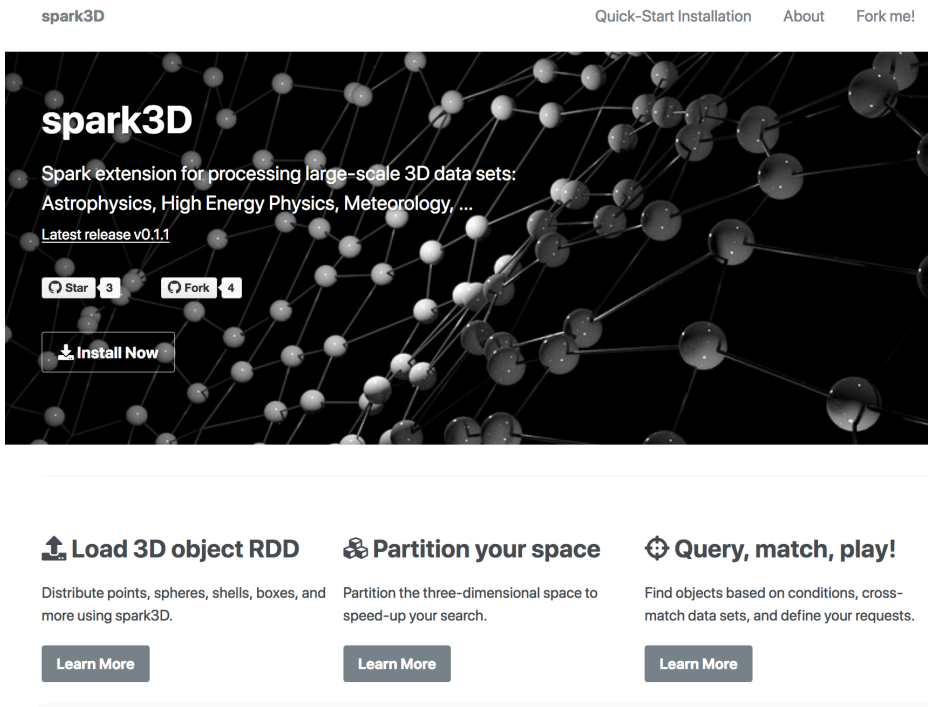
- Infrastructures:
  - 9 machines, 162 coeurs total, 308 GB RAM total, 35 TB stockage total.
  - Actuellement: 3-4 utilisateurs, essentiellement autour du traitement de données astro.
  - Bande passante IO (mesurée): 2.3 GB/s en moyenne.
  - Traitement de données jusqu'à 110 GB distribuées de l'ordre de quelques secondes une fois les données mises en cache. Traitement de 1.2 TB (cache/disque) en 10 minutes.
- Développements en cours sur le cluster:
  - Nouveau connecteur Apache Spark pour l'Astrophysique (spark-fits).
  - Traitement de jeux de données tridimensionnels (spark3D).
  - Traitement et analyse fine des logs du parc informatique.

Google Summer of Code 2018

- 1 étudiant pendant 3 mois, financé par Google, pour travailler sur un projet open source.

- 2 projets au LAL (via HSF) en 2018, dont 1 sur Apache Spark

  - spark3D: Extend Apache Spark to support 3D Spatial Datasets



CONTACTS

- Contacts (LSST/SI/DI) :

ARNAULT	Christian	LAL	arnault@lal.in2p3.fr
PATEYRON	Sacha	LAL	sacha.pateyron@lal.in2p3.fr
PELOTON	Julien	LAL	peloton@lal.in2p3.fr
PHILIPPON	Guillaume	LAL	philippo@lal.in2p3.fr
PLASZCZYNSKI	Stéphane	LAL	plaszczy@lal.in2p3.fr
RAMPARISON	Adrien	LAL	ramparison@lal.in2p3.fr
RICHARD	César	DI	Cesar.Richard@u-psud.fr

Table – List of contacts

En cours, en réflexion

- Haute disponibilité : Zookeeper
- Mise en ligne déploiement par ansible
- Renforcement de l'activité 'calcul distribué' (Spark, Hadoop, HDFS, ...)
- Autres projets (analyse et corrélation de logs, ...)
- Montée en charge de l'infrastructure
- Amélioration du monitoring (ganglia, check_mk), slack pour les alertes
- Hue : Web UI pour la prise en charge de Hadoop et son écosystème. Une interface web pour Hive, Impala, HDFS, Spark, soumission des jobs ...
- Utilisation des notebooks Jupyter (Zeppelin)
- Réflexion sur le stockage : HDFS vs Ceph(FS) vs S3
- Discussion avec CERN-IT

Conclusion

- Open source
- Projet jeune mais mature (2009, UC Berkeley), innovant
- Performant pour le traitement des données volumineuses
- Grande communauté (Apache Foundation, Databricks, Hortonworks, Cern, ...)
- De plus en plus utilisé

Questions ¿

Merci pour votre attention.

Questions ¿