BIG DATA
HADOOP – HDFS- HIVE

HADOOP

- **Hadoop** est un **framework** libre et **open source** écrit en Java destiné à faciliter la création **d'applications distribuées** (au niveau du stockage des données et de leur traitement) et échelonnables (**scalables**) permettant aux applications de travailler avec des **milliers de nœuds** et des **pétaoctets de données**.

HADOOP

- Chaque nœud est constitué de **machines standard** regroupées en grappe.
- Tous les modules de Hadoop sont conçus dans l'idée fondamentale que les **pannes matérielles sont fréquentes** et qu'en conséquence elles doivent être gérées automatiquement par le framework.
- Hadoop a été inspiré par la publication de MapReduce, GoogleFS et BigTable de Google.
- Hadoop a été créé par **Doug Cutting** et fait partie des projets de la fondation logicielle **Apache depuis 2009**.

HISTORIQUE

- **En 2004**, Google publie un article présentant son algorithme basé sur des opérations analytiques à grande échelle sur un grand cluster de serveurs, le MapReduce, ainsi que son système de fichier en cluster, le GoogleFS.
- Doug Cutting, décide de reprendre les concepts décrits dans l'article pour développer sa propre version des outils en version Open Source, qui deviendra le projet Hadoop.
- Il s'inspire du doudou de son fils de cinq ans, un éléphant jaune, pour le logo ainsi que pour le nom de ce nouveau framework Java.
- **En 2006,** Doug Cutting a décidé de rejoindre Yahoo
- **En 2008**, Yahoo proposa Hadoop sous la forme d'un projet open source.
- **En 2011**, Hadoop en sa version 1.0.0 voit le jour; en date du 27 décembre 2011.
- **En 2012**, la communauté open source lance Hadoop 2.0 sponsorisé par la Apache Software Foundation
- La révolution majeure a été l'ajout de la couche YARN dans la structure de Hadoop.
- À partir de septembre 2016, la version 3.0.0-alpha1 est rendue disponible.

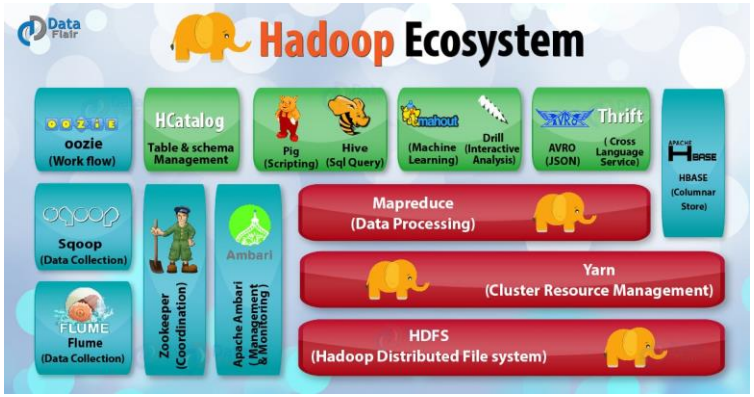
CONSTITUANTS

- Deux principales composantes
  - **Stockage** : HDFS (Hadoop Distributed File System)
  - **Traitement** : MapReduce.
- Hadoop fractionne les fichiers en **gros blocs** et les **distribue** à travers les nœuds du **cluster**.
- Pour traiter les données, Hadoop transfère **le code à chaque nœud** et chaque nœud traite **les données dont il dispose**.
- Cela permet de traiter l'ensemble des données plus rapidement et plus efficacement que dans une architecture supercalculateur
  - une architecture supercalculateur repose sur un système de fichiers parallèle où les calculs et les données sont distribués via les réseaux à grande vitesse.

MODULES

- Le framework Hadoop de base se compose des modules suivants
  - **Hadoop Common** : collection d'utilitaires et de bibliothèques communs qui prennent en charge d'autres modules Hadoop
  - **Hadoop Distributed File System (HDFS)** : Système distribué de gestion de fichiers
  - **Hadoop YARN (Yet Another Resource Negotiator)** : Gestion des ressources du Cluster
  - **Hadoop MapReduce** : traitement des données

ECOSYSTEME HADOOP



ECOSYSTEME HADOOP

- **Apache Pig**
  - Plateforme haut niveau pour la création de programme MapReduce.
  - Le langage de cette plateforme est appelé le **Pig Latin**.
  - Pig Latin s'abstrait du langage de programmation Java MapReduce et se place à un niveau d'abstraction supérieur, similaire à celle de SQL pour systèmes SGBDR
  - Initialement développé chez Yahoo Research dans les années 2006 pour les chercheurs qui souhaitaient avoir une solution ad-hoc pour créer et exécuter des jobs map-reduce sur d'importants jeux de données.
  - En 2007, il a été transmis à l'Apache Software Foundation.

ECOSYSTEME HADOOP

- **Apache Hive**
  - Infrastructure d'entrepôt de données intégrée sur Hadoop permettant l'analyse, le requêtage via un langage proche syntaxiquement de SQL ainsi que la synthèse de données
  - Prend en charge l'analyse des grands ensembles de données stockées dans Hadoop HDFS.
  - Il fournit un langage similaire à SQL appelée HiveQL; de manière transparente il convertit les requêtes en map/reduce et jobs Spark.
  - Pour accélérer les requêtes, il fournit des index, y compris bitmap indexes,

ECOSYSTEME HADOOP

- **Apache Hbase**
  - Système de gestion de base de données non-relationnelles distribué, écrit en Java, disposant d'un stockage structuré pour les grandes tables.
  - Inspirée des publications de Google sur BigTable.
  - Base de données orientée colonnes.
  - Basées sur une architecture maître/esclave, capable de gérer d'énormes quantités d'informations (plusieurs milliards de lignes par table)

ECOSYSTEME HADOOP

- **Apache Phoenix**
  - Un moteur de base de donnée relationnel open source, massivement parallèle, supportant OLTP pour Hadoop utilisant Apache HBase comme support de sauvegarde.
  - Fournit un driver JDBC pilote qui cache la complexité du stockage noSQL permettant aux utilisateurs de créer, supprimer et modifier des tables, des vues, des index et des séquences SQL; insérer et supprimer des lignes individuellement et en masse par le biais de SQL2.
  - Compile les requêtes et les autres états en natif noSQL

ECOSYSTEME HADOOP

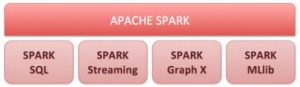
- **Apache Spark**
  - Framework open source de calcul distribué.
  - Ensemble d'outils et de composants logiciels structurés selon une architecture définie.
  - Développé à l'université de Californie à Berkeley par AMPLab3, Spark est aujourd'hui un projet de la fondation Apache.
  - Cadre applicatif de traitements big data pour effectuer des analyses complexes à grande échelle.
  - L'origine son développement est une solution pour accélérer le traitement des systèmes Hadoop
  - En 2014, Spark a gagné le **Daytona GraySort Contest**
  - Objectif : trier 100 To de données le plus rapidement possible.
  - Ce record était préalablement détenu par Hadoop.
  - Spark a utilisé 206 machines (temps d'exécution de 23 minutes), Hadoop avait lui utilisé 2100 machines (temps d'exécution 72 minutes).
  - La puissance de Spark fut démontrée en étant 3 fois plus rapide et en utilisant approximativement 10 fois moins de machines.

SPARK

- Spark exécute la totalité des opérations d'analyse de données en mémoire et en temps réel. Il s'appuie sur des disques seulement lorsque sa mémoire n'est plus suffisante.
- À l'inverse, avec Hadoop les données sont écrites sur le disque après chacune des opérations
- Ce travail en mémoire permet de réduire les temps de latence entre les traitements, ce qui explique une telle rapidité.

OUTILS SPARK

- **Spark SQL :** permet d'exécuter des requêtes en langages SQL pour charger et transformer des données.
- **Spark Streaming** : offre à son utilisateur un traitement des données en flux
- **Spark Graph X** : permet de traiter les informations issues de graphes
- **Spark Mllib :** bibliothèque d'apprentissage automatique, apparu dans la version 1.2 de Spark, qui contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le clustering, le filtrage collaboratif et la réduction de dimensions, en plus des primitives d'optimisation sous-jacentes



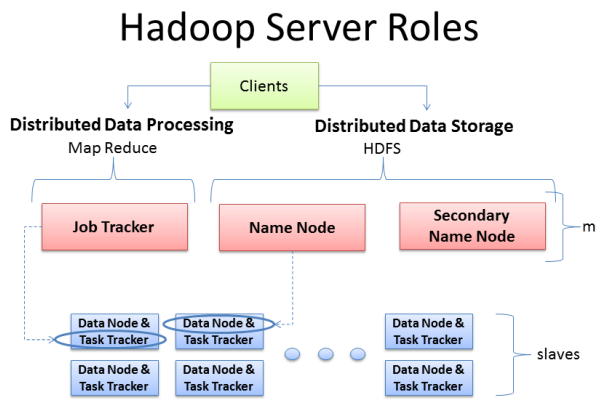
ECOSYSTEME HADOOP

- **Apache ZooKeeper**
  - Logiciel open source pour la gestion de configuration pour systèmes distribués.
  - L'architecture de ZooKeeper supporte une haute disponibilité grâce à des services redondants.
- **Apache Impala**
  - Moteur de requêtes SQL open source de Cloudera (MPP) pour les données stockées dans des cluster d'ordinateurs exécutant Apache Hadoop
  - Impala est favorisée par les analystes et les data scientists pour effectuer des analyses sur des données stockées dans Hadoop via des outils de SQL ou des outils de business intelligence.
- **Apache Flume**
  - Logiciel destiné à la collecte et à l'analyse de fichiers de log. L'outil est conçu pour fonctionner au sein d'une architecture informatique distribuée et ainsi supporter les pics de charge

ECOSYSTEME HADOOP

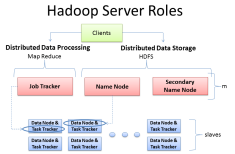
- **Apache Sqoop**
  - Interface en ligne de commande pour transférer des données entre des bases de données relationnelles et Hadoop.
  - Il prend en charge le chargement différentiels d'une seule table ou d'une requête SQL
  - Les imports peuvent être utilisés pour remplir les tables dans Hive ou HBase3.
  - Les Exportations peuvent être utilisés pour mettre les données de Hadoop dans une base de données relationnelle.
- **Apache Oozie**
  - Logiciel servant à l'ordonnancement de flux dédié au logiciel Hadoop. Il est implémenté comme une application Web Java,
- **Apache Storm**
  - Apache Storm est un framework de calcul de traitement de flux distribué, écrit principalement dans le langage de programmation Clojure.

RÔLES DES MACHINES HADDOP



RÔLES DES MACHINES HADDOP

- **Trois principaux rôles**
  - Machines Clientes
  - Nœuds Maîtres (Master Node)
  - Nœuds Esclaves (Slave Node)



- **Les nœuds maîtres** supervisent les deux éléments fonctionnels clés qui composent Hadoop: le stockage de nombreuses données (HDFS) et l'exécution de calculs parallèles sur toutes ces données (Map Reduce).
- **Le Name Node** supervise et coordonne la fonction de stockage de données (HDFS)
- **Le Job Tracker** supervise et coordonne le traitement parallèle des données à l'aide de Map Reduce.

RÔLES DES MACHINES HADDOP

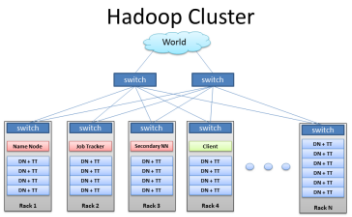
- **Les nœuds esclaves** constituent la grande majorité des machines et font tout le sale travail de stockage des données et d'exécution des calculs.
- **Chaque esclave** exécute à la fois un **Data Node démon** et **Task Tracker démon** qui communiquent avec et reçoivent des instructions de leurs nœuds maîtres.
- **Le démon Task Tracker** est esclave du Job Tracker, **le démon Data Node** est esclave du Name Node.
- Hadoop est installé sur les machines clientes avec tous les paramètres du cluster, mais n'est ni maître ni esclave.

MACHINE CLIENTE

- Rôle de **la machine cliente**
  - **Charger des données dans le cluster**,
  - Soumettre des travaux Map Reduce décrivant comment ces données doivent être traitées
  - Récupérer ou d'afficher les résultats du travail une fois terminé.
- Dans les petits clusters (~ 40 nœuds), **un seul serveur physique** peut jouer plusieurs rôles, tels que Job Tracker et Name Node.
- Pour des clusters moyens ou grands, chaque rôle opère sur une seule machine serveur.

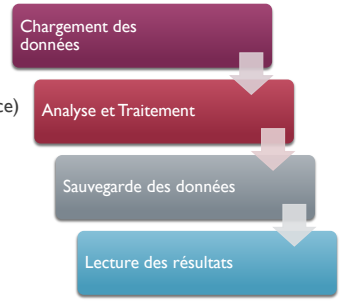
ARCHITECTURE TYPIQUE D'UN CLUSTER HADOOP

- Serveurs en Rack connectés à un Switch
- Les switchs sont connectés à un autre niveau de switch qui assure la connexion vers l'extérieur
- La majorité des serveurs seront des nœuds esclaves avec beaucoup de stockage sur disque local et des quantités modérées de CPU et de DRAM.
- Certaines des machines seront des nœuds maîtres qui pourraient avoir une configuration légèrement différente favorisant plus de DRAM et de CPU, moins de stockage local



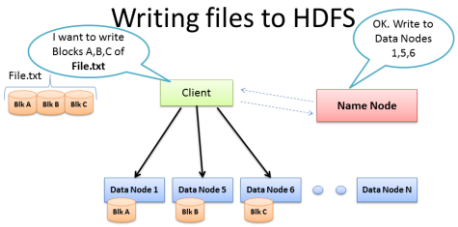
PROCESSUS DE TRAITEMENT

1. Charger les données dans le cluster
   - Ecriture des fichiers sources sur HDFS
2. Analyser et traiter les données (Map-Reduce)
3. Sauvegarder les résultats sur le cluster
   - Ecriture des résultats dans HDFS
4. Lecture des résultats à partir du Cluster
   - Lecture des blocs HDFS



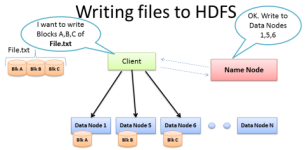
PROCESSUS DE TRAITEMENT

- Le client consulte le Name Node
- Le client écrit un bock directement sur un Data Node
- Les Data Nodes répliquent les données
- Répéter le cycle pour les blocs suivants



PROCESSUS DE TRAITEMENT

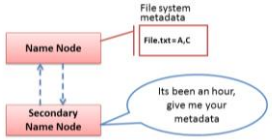
- Sans données le cluster ne sert à rien
- Le client divise le grand fichier **File.txt** en blocs de plus petites taille et place les blocs sur les machines du cluster
- + blocs= + de machines en parallèle
- **Risque de pannes** : réplication de chaque bloc sur plusieurs machine (3 par défaut)
- Paramètre **dfs.replication** à modifier dans hdfs-site.xml
- Hadoop vérifie que deux copies du même bloc sont sauvegardées dans le même rack et une copie dans un autre rack (pannes matérielles)



Le client divise **File.txt** en 3 blocs. Pour chaque bloc, le client consulte le **Name Node** et reçois 3 **Data Node** qui reçoivent les 3 blocs.
Le client écrit les blocs sur les 3 **Data Nodes**, ce derniers répliquent les blocs.
Le **Name Node** ne manipule pas les données mais donne les chemins et garde une trace des méta-données

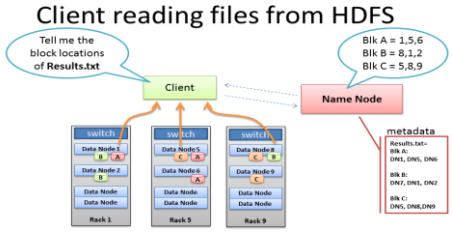
NAME NODE SECONDAIRE

- Dans Hadoop, pour chaque Name Node on créé un serveur Name Node secondaire (NNS)
- Le NNS se connecte périodiquement au Name Node (chaque heure par défaut)
- Récupère une copie des méta-données dans le Name Node ainsi que des fichiers utilisés pour stocker les métadonnées
- Le NNS combine ces informations et retourne une copie au Name Node.
- Si le Name Node crash, les méra-données sauvegardées dans le NNS peuvent être utilisées pour récupérer le Name Node



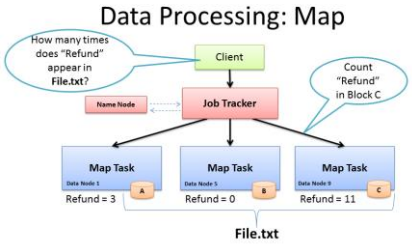
LECTURE DU RÉSULTAT

- Lorsque le client veut récupérer le résultat
- Consulte le Name Node et demande les emplacements de blocs du fichiers
- Le NN renvoie une liste de data Node pour chaque bloc
- Le Client choisit un Data Node pour lire chaque bloc



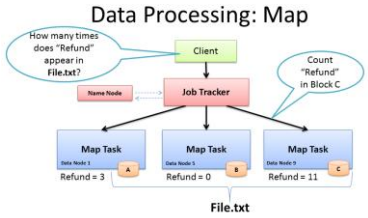
LE TRAITEMENT - MAP

- Une fois les données dans les DN, on peut lancer un traitement Map-Reduce
- Le client envoie un job map-reduce au JobTracker **'Combien de fois le mot Refund se trouve dans mon fichier'.**
- Le Job Tracker consulte le Name Node pour savoir quels nœuds de données ont des blocs de File.txt



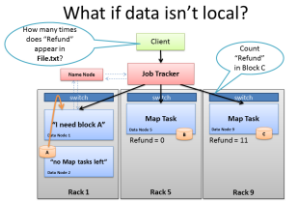
LE TRAITEMENT - MAP

- **Le Job Tracker** fournit ensuite au Task Tracker le code java de la fonction map pour l'exécuter localement
- **Le Task Tracker** lance le mappeur et supervise la progression
- Le Task Tracker fournit l'état des tâches au Job Tracker.
- À la fin de chaque tâche MAP , chaque nœud stocke le résultat de son calcul local dans un stockage local temporaire (données intermédiaires).
- Envoyer ces données intermédiaires sur le réseau à un nœud exécutant une tâche de réduction pour le calcul final.



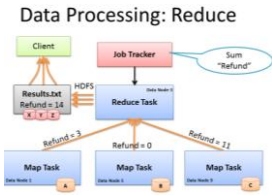
DONNÉES NON LOCALE

- Bien que Job Tracker essaie toujours de sélectionner des nœuds avec des données locales pour une tâche map, il peut ne pas toujours être en mesure de le faire.
- Une des raisons pourrait être que tous les nœuds avec des données locales ont déjà trop d'autres tâches en cours d'exécution et ne peuvent plus accepter.
- Dans ce cas, le Job Tracker consultera le nœud de nom dont la connaissance du rack peut suggérer d'autres nœuds dans le même rack.
- Le Job Tracker attribuera la tâche à un nœud dans le même rack, et lorsque ce nœud va trouver les données dont il a besoin, le nœud de nom lui demandera de récupérer les données d'un autre nœud dans son rack, en tirant parti de la haute bande passante de la commutation en rack.



REDUCE

- Rassembler toutes ces données intermédiaires pour les combiner et les distiller pour un traitement ultérieur de sorte que nous ayons un résultat final.
- Le Job Tracker démarre une tâche reduce sur l'un des nœuds du cluster et demande à la tâche reduce d'aller récupérer les données intermédiaires de toutes les tâches map terminées.
- La tâche Reduce a maintenant collecté toutes les données intermédiaires des tâches map et peut commencer la phase de calcul finale. Dans ce cas, nous additionnons simplement la somme des occurrences totales du mot «Remboursement» et écrivons le résultat dans un fichier appelé Results.txt
- La sortie du travail est un fichier appelé **Results.txt** qui est écrit sur HDFS (division du fichier en blocs, réplication du pipeline de ces blocs, etc).
- Une fois terminée, la machine cliente peut lire le fichier Results.txt à partir de HDFS, et le travail est considéré comme terminé.



HDFS

- Le HDFS est un système de fichiers distribué, extensible et portable développé par Hadoop à partir du GoogleFS.
- Écrit en Java, il a été conçu pour stocker de très gros volumes de données sur un grand nombre de machines équipées de disques durs banalisés.
- Il permet l'abstraction de l'architecture physique de stockage, afin de manipuler un système de fichiers distribué comme s'il s'agissait d'un disque dur unique.

ARCHITECTURE