BIG DATA

3 - Hadoop

POLYTECH NANTES | Pôle Sciences et technologie Nantes Université

SOMMAIRE

INTRODUCTION

Hadoop est un framework libre open source écrit en Java destiné à faciliter la création d'applications distribuées et scalables.

Le framework Hadoop de base se compose de 4 modules:

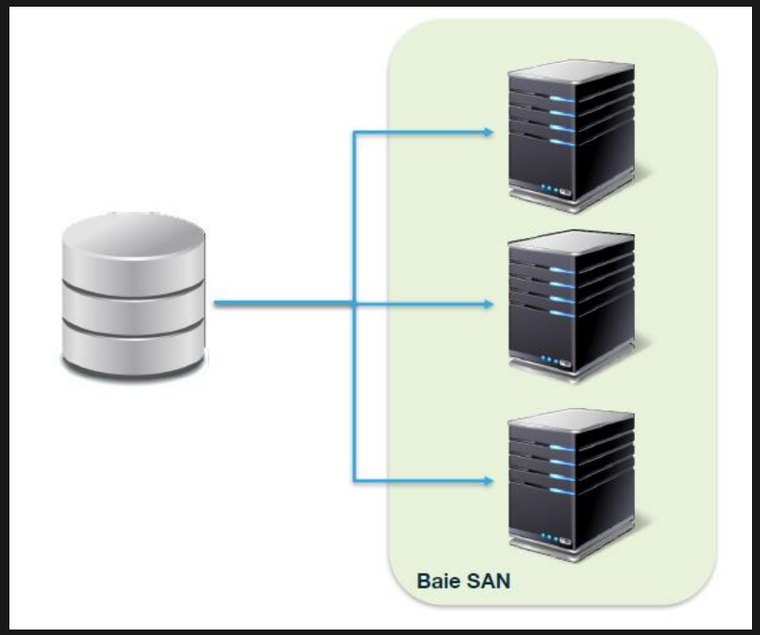
- **Hadoop Common**: ensemble de bibliothèques et utilitaires utilisés par les autres modules.
- **Hadoop Distributed File System (HDFS)**: système de fichiers distribué dédié au stockage des données.
- **Yet Another Resource Negotiator (YARN)**: permet de gérer les ressources pour les processus exécutés sur la plateforme.
- **MapReduce**: framework logiciel de traitements parallèles.

Vient ensuite s'y connecter une liste de modules supplémentaires: **Spark**, **Hive**, **Hbase**, **ZooKeeper**, **Pig**, **Kafka**.

INTRODUCTION

UNE APPROCHE DIFFÉRENTE

Approche traditionnelle :

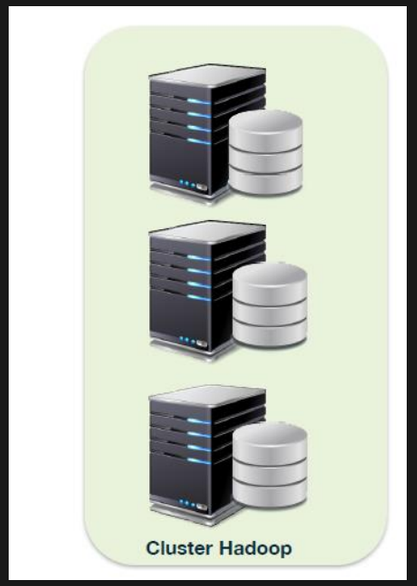


- Les données sont stockées dans une baie centralisée.
- Les données sont copiées vers les processeurs au moment du traitement.
- C'est un système efficace quand le volume de données est limité.

INTRODUCTION

UNE APPROCHE DIFFÉRENTE

Approche Hadoop :



- Une nouvelle approche:
  - Amener les programmes aux données et pas les données au programme.
- Deux principes:
  - Distribuer les données quand on les sauvegarde.
  - Traiter les données là où elles se trouvent (data locality).

INTRODUCTION

LES DISTRIBUTIONS (1/2)

CLOUDERA

- Distribution la plus diffusée
- Intégrée dans les appliances Big Data d'Oracle

- 100% Open Source
- Soutenu par des acteurs importants (Yahoo, Teradata, Microsoft)

- Amélioration de HDFS avec suppression du NameNode
- Système de fichiers vu de l'extérieur comme un NAS

En 2018 Cloudera rachète Hortonworks pour 5,2 milliards de dollars

INTRODUCTION

LES DISTRIBUTIONS (2/2)

Aujourd'hui, de nouveaux acteurs Cloud viennent proposer des alternatives aux solutions "On-Premise".

- Azure (Microsoft) : Azure Synapse Analytics
- GCP (Google) : Big Query
- AWS (Amazon) : Amazon Redshift
- Snowflake

HDFS

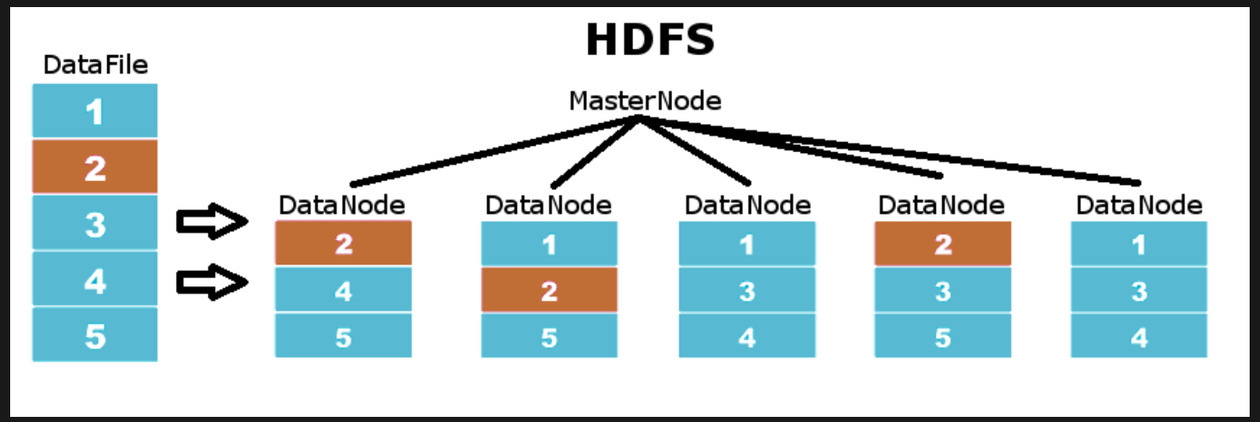
HDFS est le composant Hadoop qui permet de stocker les données (sous forme de fichiers) sur les différents noeuds qui composent son architecture.

- GFS (Google File System):
  - Système de stockage de fichiers distribué et propriétaire inventé par Google pour répondre à ses besoins croissants de stockage.
- HDFS (Hadoop Distributed File System):
  - HDFS est la version open source de GFS
  - HDFS permet de stocker tout type de fichiers, structurés ou non structurés.
  - HDFS a été conçu pour stocker de très gros volumes de données sur un grand nombre de machines équipées de disques durs banalisés.
  - HDFS 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.

HDFS

DATA DISTRIBUTION

- Un fichier est divisé en blocs:
  - Gérés par le NameNode, stockés sur les Datanodes de manière transparente pour les utilisateurs
  - Exemple: Si le paramètre de bloc est fixé à 64 Mo (paramètre par défaut), un fichier de 640 Mo sera découpé en 10 blocs.
- Le fichier est répliqué à la création sur les Datanodes
  - Exemple: Si le paramètre de réplication est 3 (paramètre par défaut), un bloc sera copié sur 3 Datanodes distincts.



HDFS

STRATÉGIE DE RÉPLICATION

- Le NameNode détermine où copier les blocs avec une stratégie d'optimisation de la bande passante:
  - Première copie sur le rack
  - Deuxième copie sur un DataNode différent et sur un autre rack
  - Troisième copie sur un DataNode sur le second rack

HDFS

STRATÉGIE D'INDEXATION

- Le NameNode monte les informations de blocs en mémoire pour accélérer les recherches:
  - Les informations d'un fichier consomment 200 octets en mémoire
  - Les informations d'un bloc consomment 200 octets en mémoire
  - Une multitude de petits fichiers consommera inutilement les ressources mémoires du NameNode et dégradera les performances globales du cluster

HDFS

FORCES

- Stockage de gros volumes:
  - Des téraoctets ou pétaoctets de données
  - Des millions de fichiers plutôt que des milliards
  - Des fichiers de plus de 100Mo plutôt que des fichiers de 100 octets
- Stockage de données temps réel (streaming):
  - Pattern "write once, read many"
- Utilisation de matériels standards:
  - Pas besoin de super ordinateurs

HDFS

FAIBLESSES

- Latence de lecture:
  - HDFS est fait pour lire de gros fichiers en continu plutôt que des petits fichiers
- Pas d'update des fichiers:
  - HDFS ne permet pas de modifier directement des fichiers
  - Si on souhaite le faire, il faut lire le fichier, le modifier puis écrire à nouveau le fichier
  - Cette limitation forte permet de garantir l'absence de verrou sur les fichiers et des performances optimales même sur des systèmes fortement distribués

HDFS

ARCHITECTURE (1/4)

NameNode : stocke toutes les métadonnées

- Les informations sur les emplacements des données sur les disques
- Les informations concernant les droits sur les fichiers
- Le nom des blocs et leur emplacement
- Le NameNode n'écrit ni ne lit directement les données (ce qui garantit une bonne scalabilité)

HDFS

ARCHITECTURE (2/4)

DataNode : stocke les blocs de données

- Les Datanodes s'exécutent sur plusieurs machines
- Les Datanodes informent en permanence le NameNode de leur bon fonctionnement

HDFS

ARCHITECTURE (3/4)

Secondary NameNode : Réplication du NameNode qui permet de redémarrer un cluster en cas d'incident sur le NameNode

- Les informations du NameNode sont copiées de manière périodique
- Le Secondary NameNode n'est pas accessible des Datanodes
- Le Secondary NameNode doit être installé sur une machine dédiée
- Depuis la version 2 de Hadoop, la reprise peut se faire à chaud (haute disponibilité)

HDFS

ARCHITECTURE (4/4)

Les clients intéragissent avec :

- Le NameNode pour obtenir l'adresse des blocs de données à des fins de lecture/écriture
- Les Datanodes pour lire/écrire les blocs de données

HDFS

USAGE

Les méthodes d'interaction avec HDFS:

- API Java
- Protocole réseau (HTTP, FTP, WebHDFS, etc.)
- Protocole propriétaire (Amazon S3, etc.)
- Ligne de commande

HDFS

USAGE

L'accès en ligne de commande permet de manipuler les fichiers (copie, déplacement, etc.) ainsi que les droits sur les fichiers.

Une commande standard HDFS ressemble (depuis la version 2 d'Hadoop) à:

```
hdfs dfs -command arg
```

HDFS

USAGE

```
# Creation d'un repertoire
hdfs dfs -mkdir nom_du_repertoire
# Lister le contenu d'un repertoire
hdfs dfs -ls nom_du_repertoire
# Envoyer un fichier sur HDFS (Local => HDFS)
hdfs dfs -put fichier_local nom_du_repertoire_cible
# Recuperer un fichier sur HDFS (HDFS => Local)
hdfs dfs -get fichier_source
# Copier un fichier (HDFS => HDFS)
hdfs dfs -cp fichier_source fichier_destination
# Deplacer un fichier (HDFS => HDFS)
hdfs dfs -mv fichier_source nom_du_repertoire_cible
# Supprimer un fichier
hdfs dfs -rm fichier_source
# Visualiser un fichier
hdfs dfs -cat fichier_source
```

HDFS

USAGE

```
# Verifier l'integrite de HDFS
hdfs fsck nom_du_repertoire
# Lister le contenu d'un repertoire
hdfs dfsadmin -report
```

HDFS

QUIZZ

- Sachant que:
  - Les informations d'un fichier consomment 200 octets en mémoire.
  - Les informations d'un bloc consomment 200 octets en mémoire.
  - Le paramétrage Hadoop est celui par défaut:
    - Taille de bloc par défaut: 64Mo
    - Réplication par défaut: 3
- Quelle mémoire est nécessaire au NameNode pour stocker:
  1. 1 fichier de 1 Go ?
  2. 1024 fichiers de 1 Mo ?

HDFS

QUIZZ - SOLUTIONS

- 1 fichier de 1 Go ?
  - Pour les fichiers : 200 octets
  - Pour les blocs :
    Nombre de blocs : 1024/64 = 16
    16 * 3 * 200 = 9600 octets
  - **Total : 9800 octets**
- 1024 fichiers de 1 Mo ?
  - Pour les fichiers : 1024 * 200 = 204800 octets
  - Pour les blocs : 1024 * 3 * 200 = 614400 octets
  - **Total : 819200 octets**

HDFS

LES ALTERNATIVES CLOUD

- AWS : S3
- Azure : ADLS
- GCP : Cloud Storage

HDFS

LES TYPES DE DONNÉES

Type	Orientation	Splittable	Compressable	Hierarchical	Notes
Plain text	Row	Yes	Yes (expensive)	No	Deal with encoding
XML / JSON	Row	No (expensive)	Yes	Yes	Very difficult to split without cheating.
Sequence File	Row	Yes	Yes (block, record)	Yes (expensive)	Uses handwritt en Hadoop-style serialization. Fast/customizable but harder to develop/maintain (especially for Hierarchical data).
MapFile	Row	Yes	Yes	Yes (expensive)	Ordered, indexed SequenceFile for fast lookups.
Avro	Row	Yes	Yes	Yes	Widely used. Efficient binary layout of structured data. JSON schema embedded with data.
ORC	Column	Yes	Yes	Yes	Hive-initiated. Only accessible via Hive or HCatalog
Parquet	Column	Yes	Yes	Yes	Hive-initiated. Connectors available in all frameworks

HDFS

LES TYPES DE DONNÉES RECOMMANDÉS

Utiliser les formats optimités : Avro ou Parquet

HDFS

LES TYPES DE DONNÉES RECOMMANDÉS

Stockage colonne ou ligne :

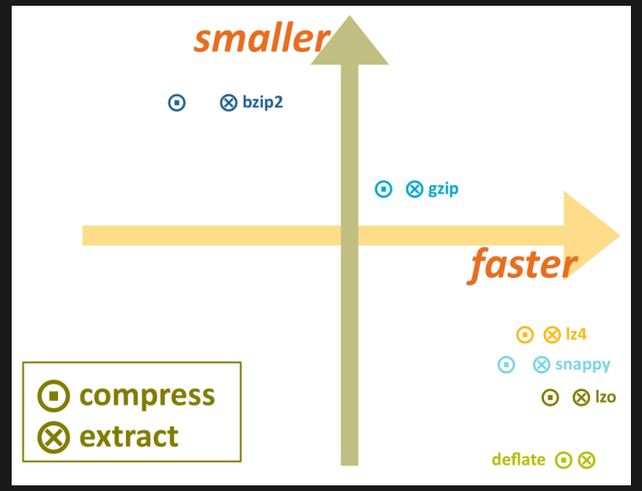
- Columnar data stores are good when not all columns for a record are needed (like aggregation queries)
- Largely reduces seek/scan time
- Usually split on chunks of rows, then organized by columns
- Optional indexes and aggregate statistics of rows in a chunk
- Columnar storage have a better compression ratio and is more suited on very high volume requirements
- Prefer Parquet over ORC, thanks to native compatibility with almost all the ecosystem whereas ORC is only for Hive / Hcatalog

HDFS

LA COMPRESSION DES DONNÉES

- Prefer Snappy or LZO (if LZO is correctly installed by Sysadmins)
- Forbid unsplittable formats on HDFS (gzip), it will limit to 1 mapper / file, highly impacting performance if files > 64MB

Compression format	Tool	Algo	Extension	Splittable ?
DEFLATE	compress	DEFLATE	.deflate/.Z	no
gzip	gzip	DEFLATE	.gz	no
bzip2	bzip2	bzip2	.bz2	yes
LZO	lzop	LZO	.lzo	yes (if encoded by records)
LZ4	n/a	LZ4	.lz4	no
Snappy	n/a	Snappy	.snappy	yes (if encoded by records)



MAPREDUCE

DÉFINITION

MapReduce permet de paralléliser un traitement sur les différents noeuds qui composent le cluster Hadoop.

Les deux phases principales de chaque traitement MapReduce sont:

- Map: permet aux différents noeuds du cluster de distribuer leur travail.
- Reduce: permet de réduire la forme finale des résultats des noeuds en un seul résultat.

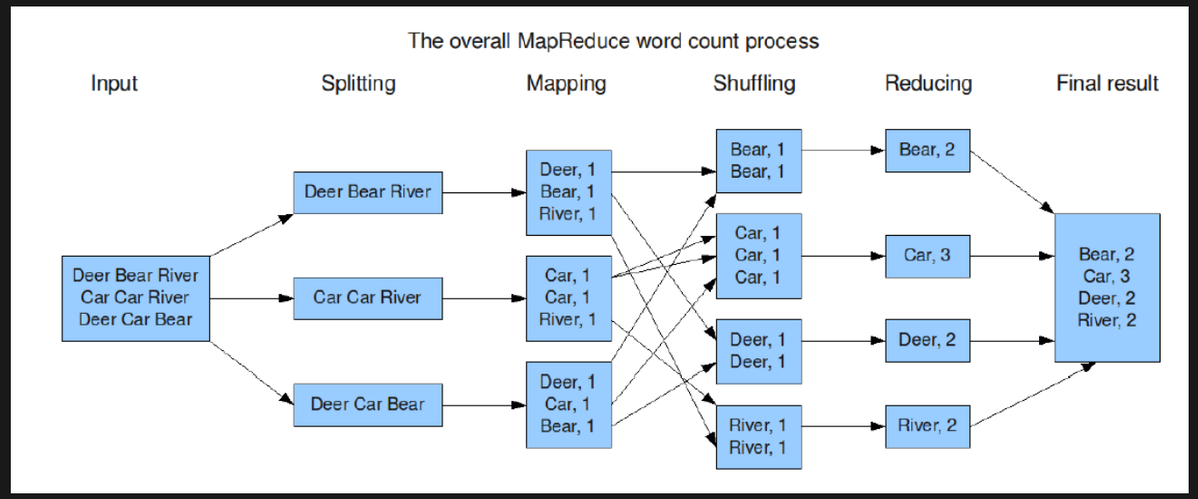
MAPREDUCE

ARCHITECTURE

- JobTracker aussi appelé "noeud maître":
  - Divise les blocs de données en bloc de même taille
  - Distribue les opérations sur les différents TaskTracker
  - Échange avec le NameNode sur l'emplacement des données
- TaskTracker aussi appelé "noeud esclave":
  - Contient les blocs de données et les répliques
  - Réalise le traitement
  - Notifie le JobTracker de l'avancement de sa tâche (heartbeat)

MAPREDUCE

EXEMPLE DE TRAITEMENT AVEC MAPREDUCE



- **Input**: Reception d'un fichier en entrée
- **Splitting**: Division des données en blocs de même taille
- **Mapping**: Exécution d'une fonction sur chaque bloc
- **Shuffling**: Trie des données par clé
- **Reducing**: Concaténation des résultats par clé
- **Final result**: Aggrégation des résultats

MAPREDUCE

FORCES

- Scalabilité:
  - Peut travailler sur un large cluster de machines
  - Exécute les tâches en parallèle
- Interopérabilité:
  - Plusieurs bibliothèques disponibles: C, C++, Java, Python, etc.
- Gestion des erreurs:
  - Duplique les tâches sur plusieurs noeuds en cas de perte d'un noeud du cluster
  - Relance automatiquement la tâche sur un nouveau noeud dans le cas où le premier ne répond plus

MAPREDUCE

FAIBLESSES

- Performance en milieu hétérogène:
  - Ne prend pas en charge les clusters dont les machines n'ont pas les mêmes caractéristiques matérielles
  - Problème de data locality dans un milieu hétérogène
- Latence:
  - Batch processing uniquement
  - Traitement en temps réel impossible
- Mise en cache:
  - Lit et écrit les données uniquement sur disque
  - Pas de mise en cache des résultats intermédiaires
  - Pas de cascade de traitement `in-memory' possible
  - Nécessite l'écriture et la relecture des données à chaque étape

MAPREDUCE

QUIZZ

- Sachant que:
  - Le traitement WordCount prend 1 minute pour traiter un fichier de 64Mo sur un cluster avec 3 noeuds.
  - Le paramétrage Hadoop est celui par défaut:
    - Taille de bloc par défaut: 64Mo
    - Réplication par défaut: 3
  - On considère que le temps lié à la phase de Map et celle de Reduce est négligeable.
  - Les données sont uniformément réparties sur le cluster.
- Combien de temps prendra le même traitement avec:
  1. un fichier de 64 Mo sur un cluster avec 6 noeuds?
  2. un fichier de 192 Mo sur un cluster avec 3 noeuds?
  3. un fichier de 192 Mo sur un cluster avec 6 noeuds?
  4. un fichier de 384 Mo sur un cluster avec 3 noeuds?
  5. un fichier de 384 Mo sur un cluster avec 6 noeuds?

MAPREDUCE

QUIZZ - SOLUTIONS

Combien de temps prendra le même traitement avec:

1. un fichier de 64 Mo sur un cluster avec 6 noeuds? => 1 minute.
2. un fichier de 192 Mo sur un cluster avec 3 noeuds? => 1 minute.
3. un fichier de 192 Mo sur un cluster avec 6 noeuds? => 1 minute.
4. un fichier de 384 Mo sur un cluster avec 3 noeuds? => 2 minutes.
5. un fichier de 384 Mo sur un cluster avec 6 noeuds? => 1 minute.

YARN

DÉFINITION

YARN : Yet Another Resource Negociator est le composant qui gère les ressources du cluster.

Il combine un gestionnaire de ressources, des coordinateurs d'application et des agents pour surveiller les opérations de traitement des différents noeuds du cluster.

Yarn a été conçu pour prendre en charge différents frameworks de traitement:

- MapReduce (MR2)
- Spark
- etc.

YARN

DATA LOCALITY

Le principe de **Data Locality** consiste à déplacer le programme au plus près de la donnée au lieu de déplacer la donnée au travers du réseau.

Conséquences :

- Pour assurer de bonnes performances du cluster, il faut s'assurer que les worker nodes disposent chacun d'un service DataNode et d'un service NodeManager.
- En cas d'évolution du cluster (ajout d'un ou plusieurs noeuds), il faudra également penser à redistribuer les données sur l'ensemble des noeuds (opération de balancing).

YARN intéragit énormément avec le NameNode HDFS pour respecter le principe de Data Locality.

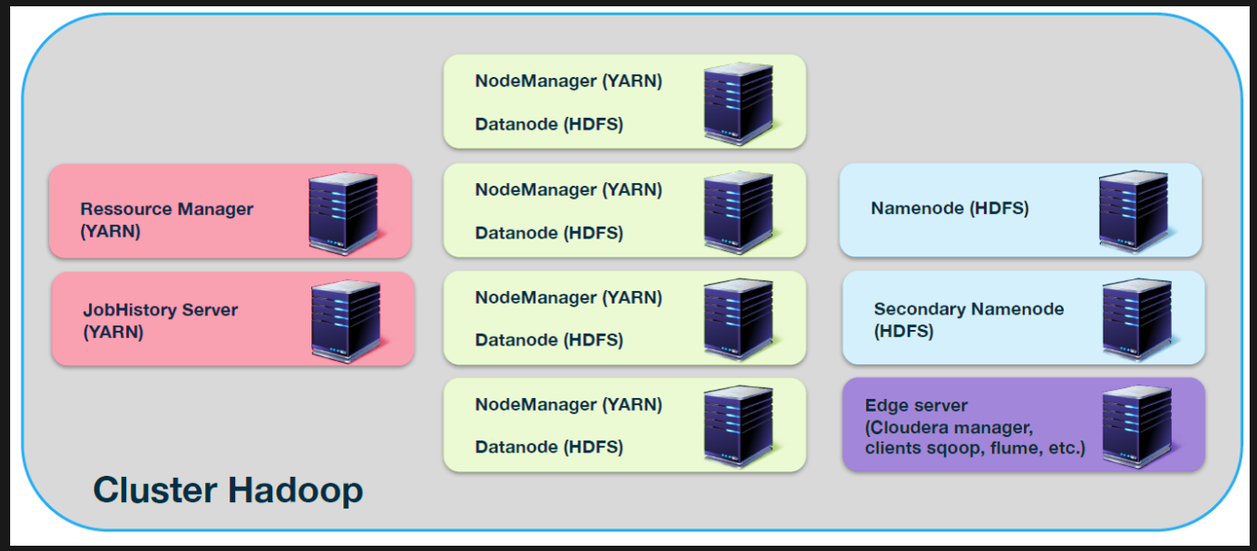
YARN

ARCHITECTURE

- Resource Manager:
  - Point d'entrée pour toutes les tâches YARN
  - Initialise les applications
  - Planifie l'usage des ressources sur les worker nodes
- NodeManager:
  - Démarre les processus des applications
  - Gère les ressources allouées sur le worker node
- JobHistory Server:
  - Archive les métriques et les métadonnées des traitements

YARN

CONFIGURATION TYPIQUE D'UN CLUSTER HADOOP



ECOSYSTÈME HADOOP

- Pig
- Hive
- Hbase
- ZooKeeper => à suivre
- NiFi
- Kafka
- Spark
- etc.

ECOSYSTÈME HADOOP

HAUTE DISPONIBILITÉ

- ZooKeeper

ZOOKEEPER

INTRODUCTION

Zookeeper est un logiciel open source de gestion de configuration pour systèmes distribués.

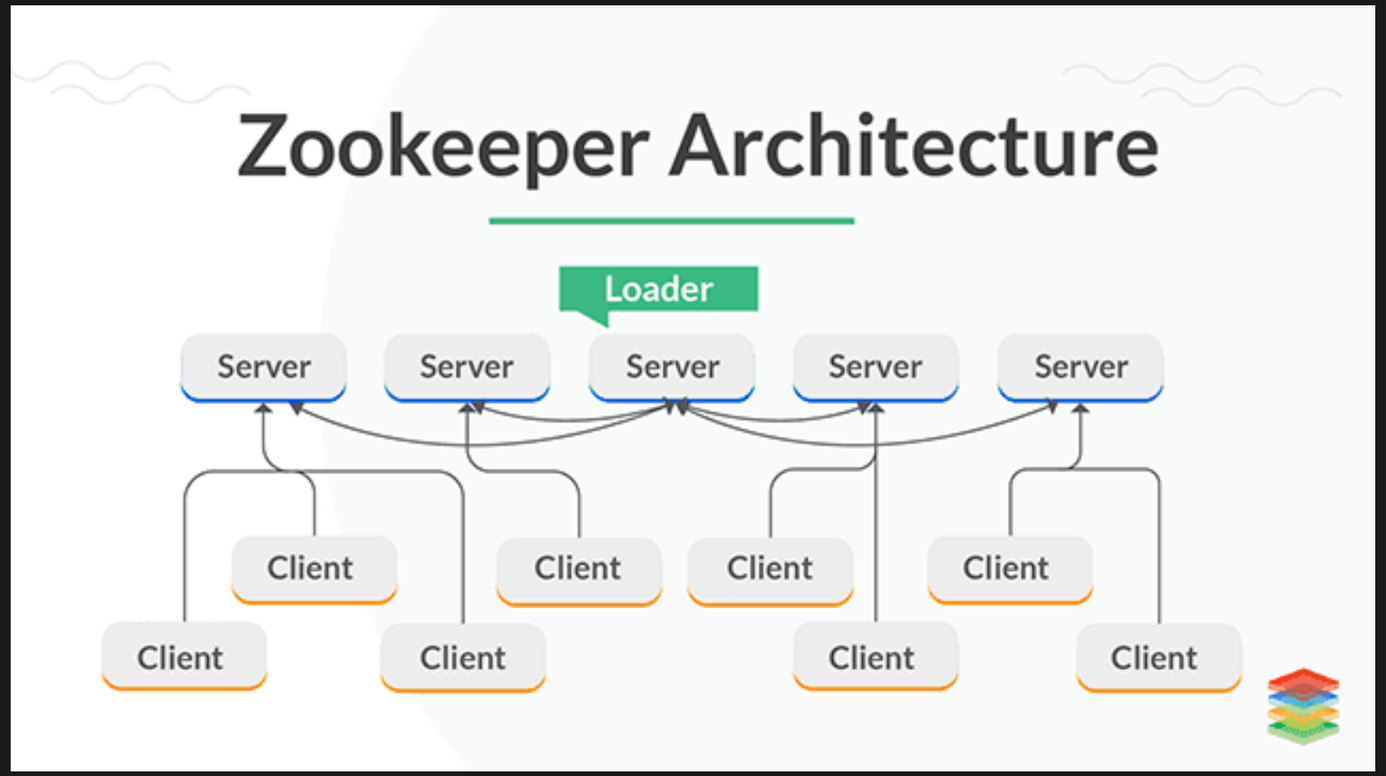
Il supporte la haute disponibilité et stocke un ensemble de clé–>valeur dont les mises à jour sont ordonnées et que les clients peuvent requêter en lecture et/ou écriture.

Cas d'utilisation :

- Gestion de configuration
- Synchronisation
- Verrous
- Election de leader
- Etc.

ZOOKEEPER

ARCHITECTURE



ZOOKEEPER

FONCTIONNEMENT

- Données distribuées et répliquées
- Tous les serveurs stocke une copie des données (in memory ou sur un filesystem local)
- Une leader est élu au démarrage
- Le leader va broadcaster les modifcations atomiques à l'ensmeble des serveurs
- Les modifications sont ordonnées
- Pas de lecture/écriture partielles

QUESTIONS ?