Mohamed Sakkari

Cours: Traitement du Big data avancé (Pig&Hive)

Objectifs

1. Appréhender le fonctionnement d'Hadoop
2. Identifier l'écosystème : quels outils pour quels usages ?
3. Manipuler les principales commandes shell d'interaction avec Hadoop
4. Émettre des requêtes SQL avec Hive et HCatalog Créer des traitements de données avec Pig

Chapitre I

Introduction à HADOOP

I. Rappel

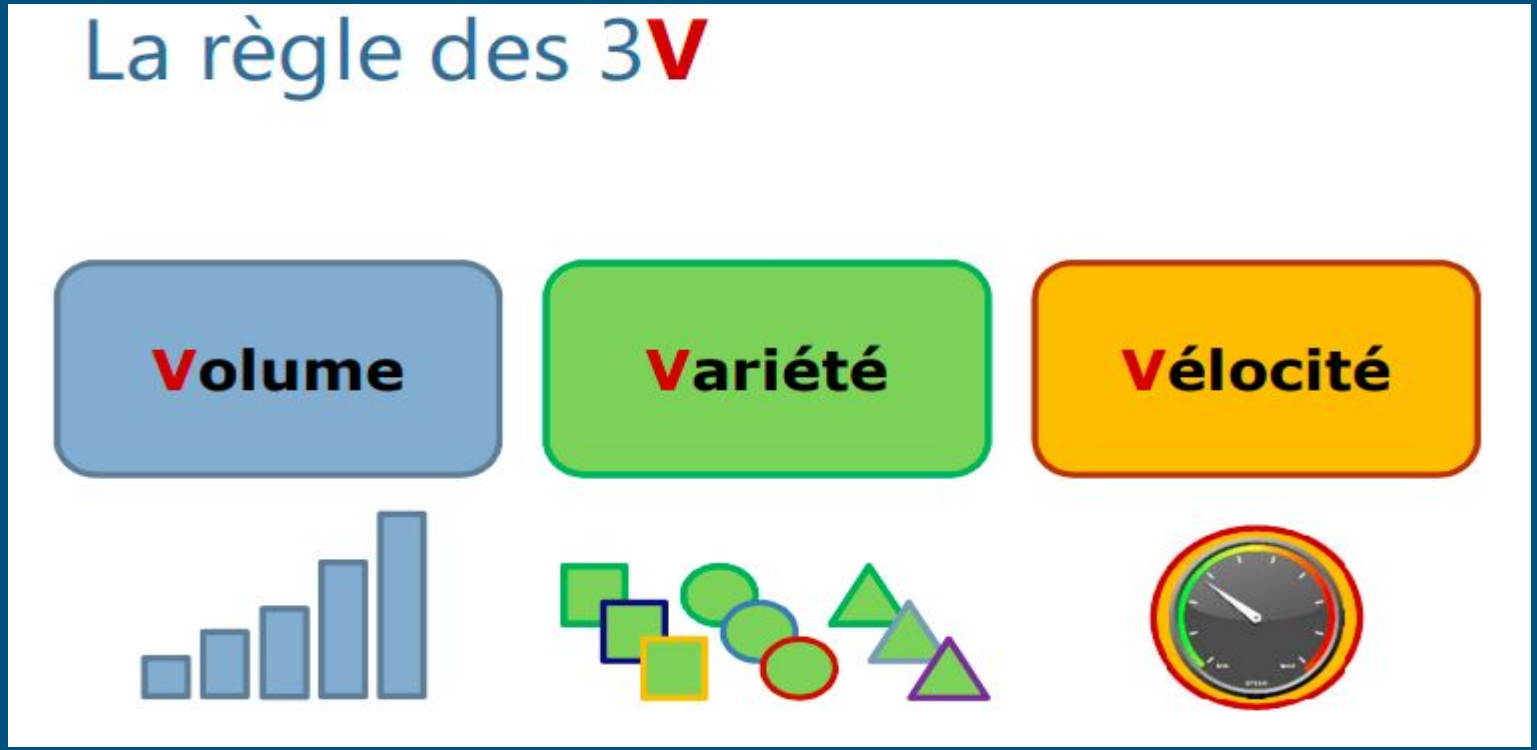
- **Préfixes multiplicatifs :**

signe	préfixe	facteur	exemple représentatif
k	kilo	10^3	une page de texte
M	méga	10^6	vitesse de transfert par seconde
G	giga	10^9	DVD, clé USB
T	téra	10^{12}	disque dur
P	**péta**	10^{15}	
E	**exa**	10^{18}	FaceBook, Amazon
Z	**zetta**	10^{21}	internet tout entier depuis 2010

I. Rappel

- Le Big Data désigne **un très grand volume de données** souvent hétérogènes qui ont **plusieurs formes et formats** (texte, données de capteurs, son, vidéo, données sur le parcours, fichiers journaux, etc.), et comprenant des formats hétérogènes : **données structurées, non structurées et semi-structurées**.
- Le Big Data a une nature complexe qui nécessite des technologies puissantes et des algorithmes avancés pour son traitement et son stockage. Ainsi, il ne peut être traité en utilisant des outils tels que les SGBD traditionnels. La plupart des scientifiques et experts des données définissent le Big Data avec **le concept des 3V.**

II. Rappel



II. Rappel

Plus 2 V

Véracité

Valeur

I. Rappel

- Distribution données et traitements : Le traitement de big data impose des méthodes particulières. Un SGBD classique, même haut de gamme, est dans l'incapacité de traiter autant d'informations.

  → Répartir les données sur plusieurs machines (jusqu'à plusieurs millions d'ordinateurs) dans des Data Centers

  – système de fichiers spécial permettant de ne voir qu'un seul espace pouvant contenir des fichiers gigantesques et/ou très nombreux (HDFS),

  – bases de données spécifiques (HBase, Cassandra, ElasticSearch).

  → Traitements du type « map-reduce » :

  – algorithmes faciles à écrire,

  – exécutions faciles à paralléliser.

I. Rappel

→ Toutes ces machines sont connectées entre elles afin de partager l'espace de stockage et la puissance de calcul.

→ Le Cloud est un exemple d'espace de stockage distribué : des fichiers sont stockés sur différentes machines, généralement en double pour prévenir une panne.

I. Rappel

→ Toutes ces machines sont connectées entre elles afin de partager l'espace de stockage et la puissance de calcul.

→ Le Cloud est un exemple d'espace de stockage distribué : des fichiers sont stockés sur différentes machines, généralement en double pour prévenir une panne.

I. Rappel

- Hadoop est un framework logiciel dédié au stockage et au traitement de larges volumes de données. Il contient de beaucoup de composants, dont :

  → HDFS un système de fichier qui répartit les données sur de nombreuses machines.

  → YARN un mécanisme d'ordonnancement de programmes de type MapReduce.

II. Le cœur de la plateforme : HDFS et YARN

- **HDFS est un système de fichiers distribué. C'est à dire :**

  →les fichiers et dossiers sont organisés en arbre (comme Unix)

  →ces fichiers sont stockés sur un grand nombre de machines de manière à rendre invisible la position exacte d'un fichier.L'accès est transparent, quelles que soient les machines qui contiennent les fichiers.

  →les fichiers sont copiés en plusieurs exemplaires pour la fiabilité et permettre des accès simultanés multiples

HDFS permet de voir tous les dossiers et fichiers de ces milliers de machines comme un seul arbre, contenant des Po de données, comme s'ils étaient sur le disque dur local.

II. Le cœur de la plateforme : HDFS et YARN

- **HDFS, Organisation des fichiers :**

  →HDFS ressemble à un système de fichiers Unix : il y a une racine, des répertoires et des fichiers. Les fichiers ont un propriétaire, un groupe et des droits d'accès.

  →Sous la racine /, il y a :

  - des répertoires pour les services Hadoop : /hbase, /tmp, /var

  - un répertoire pour les fichiers personnels des utilisateurs : /user, Dans ce répertoire, il y a aussi trois dossiers système : /user/hive, /user/history et /user/spark

  - un répertoire pour déposer des fichiers à partager avec tous les utilisateurs : /share

II. Le cœur de la plateforme : HDFS et YARN

- **Commande hdfs dfs :**

→ Apache Hadoop 2.8.4 – Overview

1. hdfs dfs -help
2. hdfs dfs -ls [noms...]
3. hdfs dfs -cat nom
4. hdfs dfs -mv ancien nouveau
5. hdfs dfs -cp ancien nouveau
6. hdfs dfs -mkdir dossier
7. hdfs dfs -rm -f -r dossier

II. Le cœur de la plateforme : HDFS et YARN

❖ **Comment fonctionne HDFS ? :**

1. Chaque fichier HDFS est découpé en blocs de taille fixe. Un bloc HDFS = 256Mo.
2. Selon la taille d'un fichier, il lui faudra un certain nombre de blocs. Sur HDFS, le dernier bloc d'un fichier fait la taille restante.
3. Les blocs d'un même fichier ne sont pas forcément tous sur la même machine.
4. Ils sont copiés chacun sur différentes machines afin d'y accéder simultanément par plusieurs processus.

II. Le cœur de la plateforme : HDFS et YARN

- **Comment fonctionne HDFS ? :**

1. Par défaut, chaque bloc est copié sur 3 machines différentes (c'est configurable).
2. Cette réplication des blocs sur plusieurs machines permet aussi de se prémunir contre les pannes.
3. Chaque fichier se trouve donc en plusieurs exemplaires et à différents endroits.

II. Le cœur de la plateforme : HDFS et YARN

❖ **Organisation des machines pour HDFS :**

1. Un cluster HDFS est constitué de machines jouant différents rôles exclusifs entre eux :

   → L'une des machines est le maître HDFS, appelé le namenode. Cette machine contient tous les noms et blocs des fichiers, comme un gros annuaire téléphonique.

II. Le cœur de la plateforme : HDFS et YARN

❖ **Organisation des machines pour HDFS :**

1. Un cluster HDFS est constitué de machines jouant différents rôles exclusifs entre eux :

   → Une autre machine est le secondary namenode, une sorte de namenode de secours, qui enregistre des sauvegardes de l'annuaire à intervalles réguliers.

II. Le cœur de la plateforme : HDFS et YARN

❖ **Organisation des machines pour HDFS :**

1. Un cluster HDFS est constitué de machines jouant différents rôles exclusifs entre eux :

   → Certaines machines sont des clients. Ce sont des points d'accès au cluster pour s'y connecter et travailler

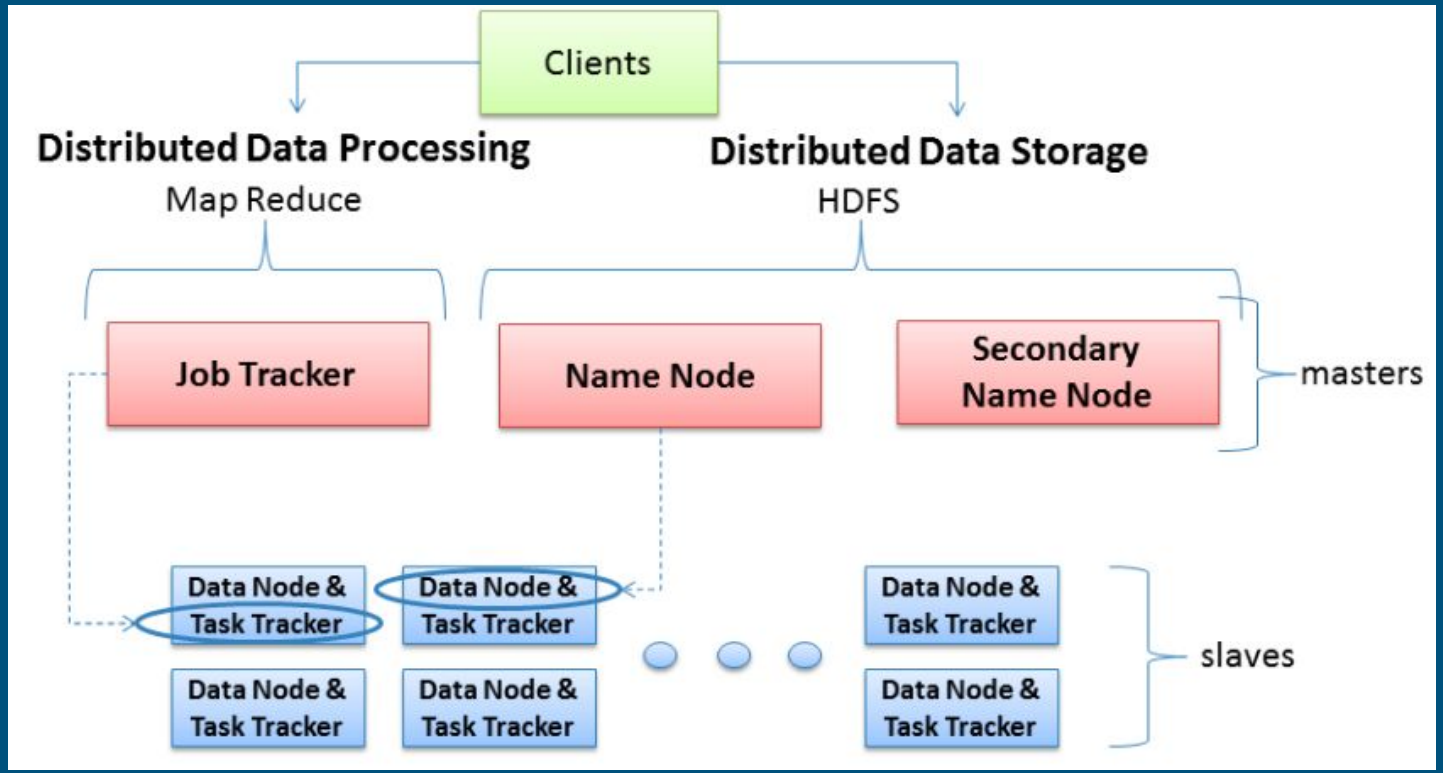
II. Le cœur de la plateforme : HDFS et YARN

❖ **Organisation des machines pour HDFS :**

1. Un cluster HDFS est constitué de machines jouant différents rôles exclusifs entre eux :

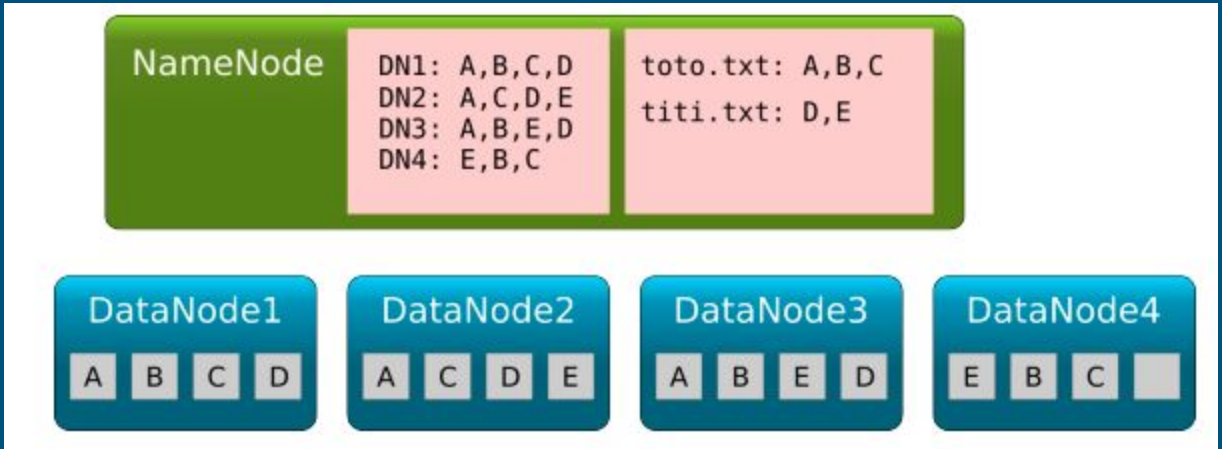
   → Toutes les autres machines sont des datanodes. Elles stockent les blocs du contenu des fichiers

II. Le cœur de la plateforme : HDFS et YARN



II. Le cœur de la plateforme : HDFS et YARN

- **exemple d'un schéma des nodes HDFS :**



Les datanodes contiennent des blocs (A, B, C. . .), le namenode sait où sont les fichiers : quels blocs et sur quels datanodes.

II. Le cœur de la plateforme : HDFS et YARN

- **exemple d'un schéma des nodes HDFS :**

→ Les datanodes contiennent des blocs.

→ Les mêmes blocs sont dupliqués (replication) sur différents datanodes, en général 3 fois. Cela assure :

-fiabilité des données en cas de panne d'un datanode,

-accès parallèle par différents processus aux mêmes données.

II. Le cœur de la plateforme : HDFS et YARN

- **exemple d'un schéma des nodes HDFS :**

Le namenode sait à la fois :

→sur quels blocs sont contenus les fichiers,

→sur quels datanodes se trouvent les blocs voulus.

On appelle cela les metadata.

Rq : Inconvénient majeur : panne du namenode = mort de HDFS, c'est pour éviter ça qu'il y a le secondary namenode. Il archive les metadata, par exemple toutes les heures.

II. Le cœur de la plateforme : HDFS et YARN

- **Mode high availability**

Comme le namenode est absolument vital pour HDFS mais unique, Hadoop propose une configuration appelée high availability dans laquelle il y a 2 autres namenodes en secours, capables de prendre le relais instantanément en cas de panne du namenode initial.

Les namenodes de secours se comportent comme des clones. Ils sont en état d'attente et mis à jour en permanence à l'aide de services appelés JournalNodes.

II. Le cœur de la plateforme : HDFS et YARN

❖ API Java pour HDFS

Hadoop propose une API Java complète pour accéder aux fichiers de HDFS. Elle repose sur deux classes principales :

1. FileSystem représente l'arbre des fichiers (file system). Cette classe permet de copier des fichiers locaux vers HDFS (et inversement), renommer, créer et supprimer des fichiers et des dossiers

   → FileSystem (Apache Hadoop Main 3.3.4 API)

II. Le cœur de la plateforme : HDFS et YARN

- **API Java pour HDFS**

Hadoop propose une API Java complète pour accéder aux fichiers de HDFS. Elle repose sur deux classes principales :

1. gère les informations d'un fichier ou dossier :
   - taille avec getLen(),
   - nature avec isDirectory() et isFile(),

   FileStatus → FileStatus (Apache Hadoop Main 3.3.4 API)

II. Le cœur de la plateforme : HDFS et YARN

- **API Java pour HDFS**

  Les deux classes FileSystem et FileStatus ont besoin de connaître la configuration du cluster HDFS, à l'aide de la classe Configuration. D'autre part, les noms complets des fichiers sont représentés par la classe Path.

II. Le cœur de la plateforme : HDFS et YARN

- **API Java pour HDFS , exemple des manipulations sur un fichier**

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.Path;

Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
Path nomcomplet = new Path("/user/etudiant1", "bonjour.txt");
FileStatus infos = fs.getFileStatus(nomcomplet);
System.out.println(Long.toString(infos.getLen())+" octets");
fs.rename(nomcomplet, new Path("/user/etudiant1","salut.txt"));
```

II. Le cœur de la plateforme : HDFS et YARN

- **API Java pour HDFS , Informations sur les fichiers**

// les importations :

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.BlockLocation;
import org.apache.hadoop.fs.Path;
```

// obtenir les informations classe HDFSinfo

```
public class HDFSinfo {

    public static void main(String[] args) throws IOException
    {
        // obtenir un object représentant HDFS
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        // nom du fichier à lire
        Path nomcomplet = new Path("apitest.txt");

        if (! fs.exists(nomcomplet)) {
            System.out.println("Le fichier n'existe pas");
        } else {

            // taille du fichier
            FileStatus infos = fs.getFileStatus(nomcomplet);
            System.out.println(Long.toString(infos.getLen()) + " octets");

            // liste des blocs du fichier
            BlockLocation[] blocks = fs.getFileBlockLocations(infos, 0, infos.getLen());
            System.out.println(blocks.length + " blocs :");
            for (BlockLocation blocloc: blocks) {
                System.out.println("\t" + blocloc.toString());
            }

        }

        // fermer HDFS
        fs.close();
    }
}
```

II. Le cœur de la plateforme : HDFS et YARN

- **API Java pour HDFS , Lecture d'un fichier HDFS**

// les importations :

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.BlockLocation;
import org.apache.hadoop.fs.Path;
```

// simple lecture d'un fichier texte classe HDFSread

```
public class HDFSread {

    public static void main(String[] args) throws IOException
    {
        // obtenir un object représentant HDFS
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        // nom du fichier à lire
        Path nomcomplet = new Path("apitest.txt");

        if (! fs.exists(nomcomplet)) {
            System.out.println("Le fichier n'existe pas");
        } else {

            // ouvrir et lire le fichier
            FSDataInputStream inStream = fs.open(nomcomplet);
            String data = inStream.readUTF();
            System.out.println(data);
            inStream.close();

        }

        // fermer HDFS
        fs.close();
    }
}
```

II. Le cœur de la plateforme : HDFS et YARN

- **API Java pour HDFS , Création d'un fichier HDFS**

// les importations :

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.BlockLocation;
import org.apache.hadoop.fs.Path;
```

// créer un fichier texte classe HDFSwrite

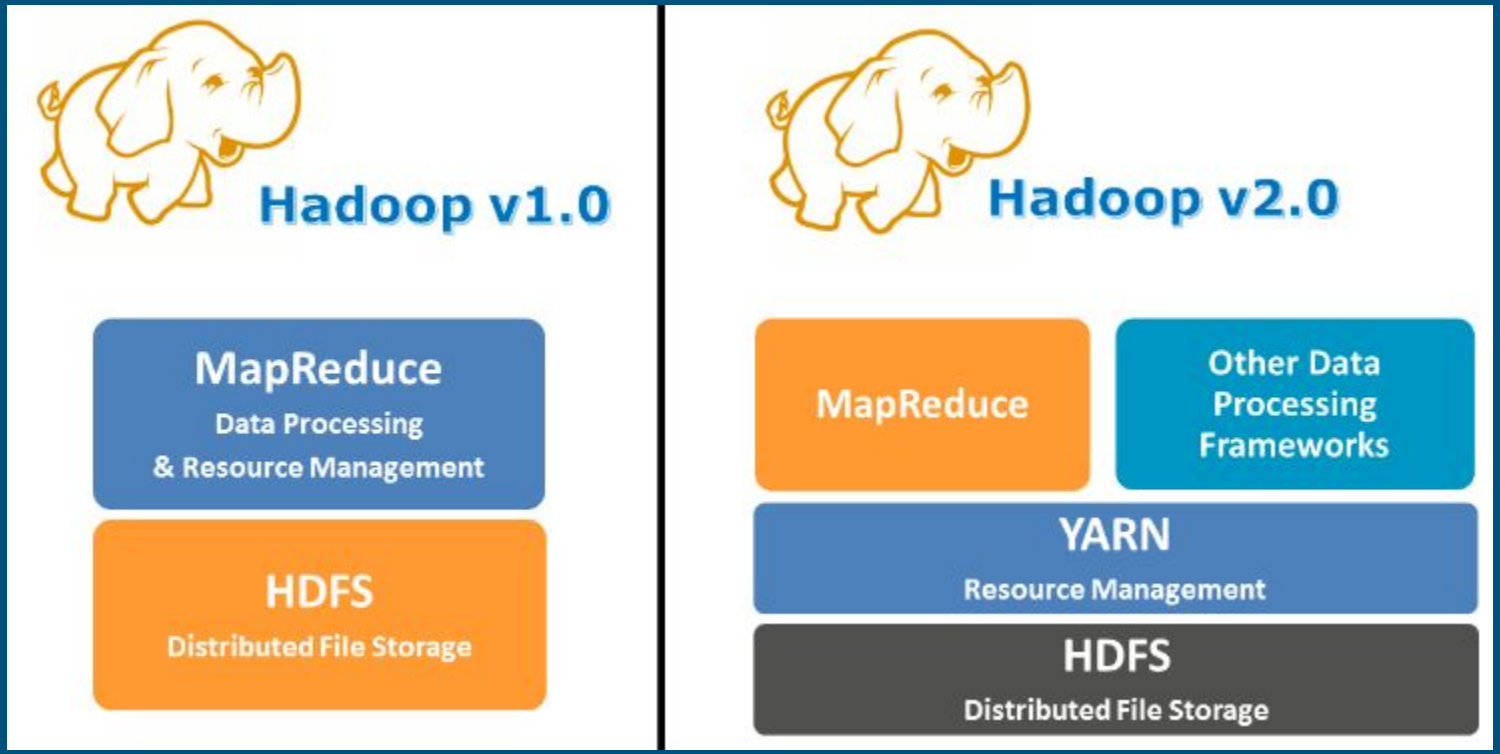
```
public class HDFSwrite {

    public static void main(String[] args) throws IOException
    {
        // obtenir un object représentant HDFS
        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);

        // nom du fichier à créer
        Path nomcomplet = new Path("apitest.txt");

        if (fs.exists(nomcomplet)) {
            System.out.println("Le fichier existe déjà.");
        } else {

            // création et écriture du fichier
            FSDataOutputStream outStream = fs.create(nomcomplet);
            outStream.writeUTF("Bonjour tout le monde !");
            outStream.close();

        }

        // fermer HDFS
        fs.close();
    }
}
```

II. Le cœur de la plateforme : HDFS et YARN

❖ **Hadoop YARN ?**

- Apache Hadoop YARN (Yet Another Resource Negotiator) est une couche de gestion des ressources dans Hadoop
- Il permet à divers moteurs de traitement de données tels que le traitement interactif, le traitement de graphes, le traitement par lots et le traitement de flux d'exécuter et de traiter les données stockées dans HDFS.
- Dans Hadoop version 1.0, également appelée MRV1 (MapReduce Version 1), MapReduce effectuait à la fois des fonctions de traitement et de gestion des ressources.

II. Le cœur de la plateforme : HDFS et YARN



II. Le cœur de la plateforme : HDFS et YARN

❖ **Hadoop YARN ?**

- IBM mentionne des limites pratiques de MRV1 d'une conception avec un cluster de 5 000 nœuds et 40 000 tâches s'exécutant simultanément. En dehors de cette limitation, l'utilisation des ressources de calcul est inefficace dans MRV1. De plus, le framework Hadoop est devenu limité uniquement au paradigme de traitement MapReduce.
- Pour surmonter tous ces problèmes, YARN a été introduit dans Hadoop version 2.0 en 2012 par Yahoo et Hortonworks.

II. Le cœur de la plateforme : HDFS et YARN

❖ **Hadoop YARN ?**

- L'idée de base derrière YARN est de soulager MapReduce en prenant en charge la gestion des ressources et la planification des tâches. YARN a commencé à donner à Hadoop la possibilité d'exécuter des tâches non MapReduce dans le cadre Hadoop.
- Avec l'introduction de YARN, l' écosystème Hadoop a été complètement révolutionné. Il est devenu beaucoup plus flexible, efficace et évolutif. Lorsque Yahoo a lancé YARN au premier trimestre de 2013, il a aidé l'entreprise à réduire la taille de son cluster Hadoop de 40 000 nœuds à 32 000 nœuds.

II. Le cœur de la plateforme : HDFS et YARN

❖ **Hadoop YARN ?**

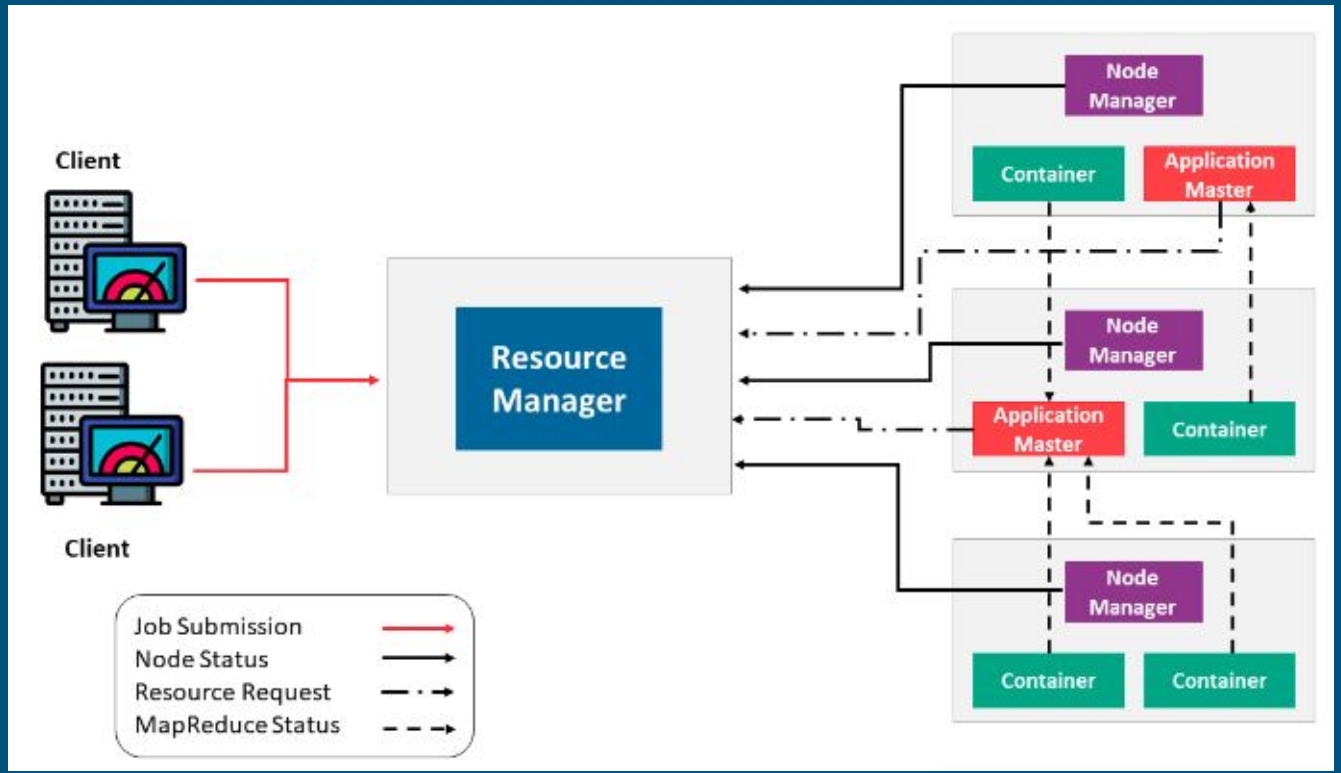
- YARN a permis aux utilisateurs d'effectuer des opérations selon les besoins en utilisant une variété d'outils tels que Spark pour le traitement en temps réel, Hive pour SQL, HBase pour NoSQL et autres.
- Outre la gestion des ressources, YARN effectue également la planification des tâches. YARN exécute toutes les activités de traitement en allouant des ressources et en planifiant des tâches.
- L'architecture Apache Hadoop YARN se compose des composants principaux suivants :

II. Le cœur de la plateforme : HDFS et YARN

- **Hadoop YARN ?**

A. Resource Manager: s'exécute sur le maître et gère l'allocation des ressources dans le cluster.
B. Node Manager:ils s'exécutent sur les esclaves et sont responsables de l'exécution d'une tâche sur chaque nœud de données.
C. Application Master:gère le cycle de vie des tâches et les besoins en ressources de chaque application. Il fonctionne avec le gestionnaire de nœuds et surveille l'exécution des tâches.
D. Container: ensemble de ressources comprenant la RAM, le processeur, le réseau, le disque dur, etc. de chaque nœud

II. Le cœur de la plateforme : HDFS et YARN



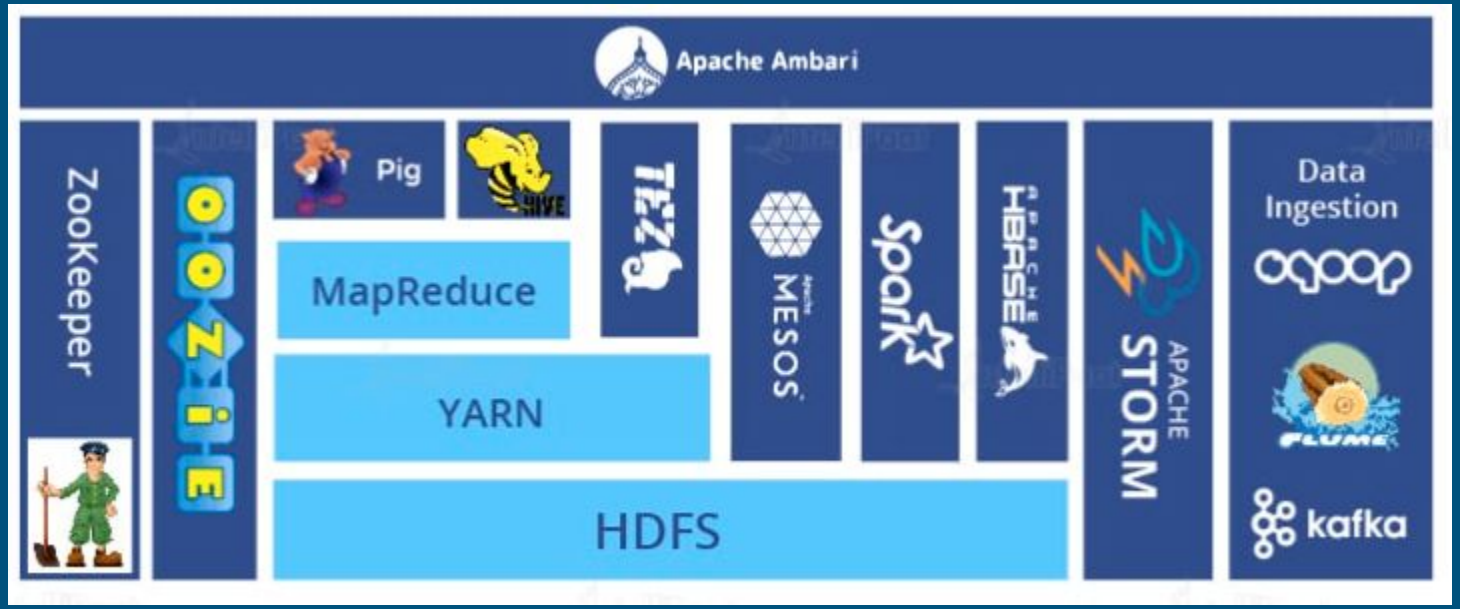
III. L’écosystème Hadoop

1. Le cœur de l'écosystème Hadoop n'est rien d'autre que les différents composants qui sont construits directement sur la plate-forme Hadoop

   → HDFS ,YARN ,MapReduce ,Apache Pig ,Apache Hive ,Apache Ambari ,Mesos ,Apache Spark ,Tez ,Apache HBase ,...

2. Cependant,il existe de nombreuses interdépendances complexes entre ces systèmes.

III. L'écosystème Hadoop



III. L’écosystème Hadoop

1. HDFS ,YARN et MapReduce font partie de Hadoop , et les autres ne sont que des projets complémentaires qui sont sortis au fil du temps et intégrés à Hadoop afin de résoudre certains problèmes spécifiques.
2. Apache Pig, c'est juste un langage de script de haut niveau qui se trouve au-dessus de MapReduce. Si vous ne voulez pas écrire de codes Java ou Python MapReduce et que vous êtes plus familier avec un langage de script qui a une syntaxe un peu de style SQL, Pig est fait pour vous.

III. L’écosystème Hadoop

1. Hive se trouve également au-dessus de MapReduce et résout un type de problème similaire à Pig, mais il ressemble plus à un SQL. Ainsi, Hive est un moyen de prendre des requêtes SQL et de faire ressembler les données distribuées quelque part sur votre système de fichiers à une base de données SQL
2. HBase : la base de données NoSQL Hadoop , Il s'agit d'une plate-forme open source, scalable horizontalement, orientée colonne, construit au-dessus du système de fichiers Hadoop. HBase est conçue sur la Bigtable de Google.

III. L’écosystème Hadoop

1. Spark est la technologie la plus intéressante de cet écosystème Hadoop. Il se trouve au même niveau que MapReduce et juste au-dessus de Mesos pour exécuter des requêtes sur vos données. Il s'agit principalement d'un moteur de traitement de données en temps réel développé afin de fournir des analyses plus rapides et faciles à utiliser que MapReduce. Spark est extrêmement rapide et fait l'objet de nombreux développements actifs. C'est une technologie très puissante car elle utilise le traitement en mémoire des données. Si vous souhaitez traiter efficacement et de manière fiable vos données sur le cluster Hadoop, vous pouvez utiliser Spark pour cela. Il peut gérer des requêtes SQL, faire du Machine Learning sur tout un cluster d'informations, gérer des flux de données, etc.

III. L’écosystème Hadoop

1. Apache Kafka est une plateforme distribuée de diffusion de données en continu, capable de publier, stocker, traiter et souscrire à des flux d'enregistrement en temps réel. Elle est conçue pour gérer des flux de données provenant de plusieurs sources et les fournir à plusieurs utilisateurs. En bref, elle ne se contente pas de déplacer un volume colossal de données d'un point A à un point B : elle peut le faire depuis n'importe quels points vers n'importe quels autres points, selon vos besoins et même simultanément.
2. …

Fin

Merci Pour Votre Attention

Chapitre II

Collecte de données et application de MapReduce

I. Analyse des flux de données dans l'entreprise

- La collecte de données et l'application de MapReduce sont deux concepts clés dans le domaine du Big Data.
- La collecte de données consiste à rassembler des données provenant de différentes sources telles que des fichiers, des bases de données, des flux de données en temps réel, des capteurs IoT, des réseaux sociaux, etc.
- Une fois que les données ont été collectées, il est souvent nécessaire de les traiter pour extraire des informations utiles.

→C'est là que l'application de MapReduce entre en jeu

I. Analyse des flux de données dans l'entreprise

- L'analyse des flux de données dans une entreprise est une méthode pour comprendre comment les données circulent à travers l'entreprise, comment elles sont collectées, stockées, transformées et utilisées pour prendre des décisions.
- Cette analyse permet de mieux comprendre les processus métier et les défis rencontrés par l'entreprise pour gérer ses données.

I. Analyse des flux de données dans l'entreprise

Les étapes clés pour réaliser une analyse des flux de données :

1. Identifier les sources de données : cela comprend toutes les sources de données internes et externes, telles que les bases de données, les fichiers, les API, les sites web, les réseaux sociaux, etc.
2. Cartographier les flux de données : pour chaque source de données, il est important de comprendre comment les données sont collectées, stockées, transformées et utilisées. Cela implique d'identifier les différents systèmes, applications et processus impliqués dans la collecte et l'utilisation des données

I. Analyse des flux de données dans l'entreprise

Les étapes clés pour réaliser une analyse des flux de données :

3. Analyser les flux de données : l'analyse peut révéler des défis tels des lacunes de données, des problèmes de qualité de données, etc.
4. Proposer des solutions : en fonction des résultats de l'analyse, il est possible de proposer des solutions pour améliorer la gestion des données dans l'entreprise. Cela peut inclure la mise en place de nouvelles technologies, la formation des employés, l'amélioration des processus métier, etc.

II. Données structurées et non-structurées

- Données structurées : les données structurées sont des données qui sont organisées dans un format prédéfini et qui peuvent être facilement stockées, recherchées et analysées.
- Les données structurées sont souvent stockées dans des bases de données relationnelles, des tableurs ou des fichiers plats.
- Les exemples de données structurées incluent les noms, les dates, les adresses, les numéros de téléphone, les prix, …

II. Données structurées et non-structurées

- Données non-structurées : les données non-structurées sont des données qui ne sont pas organisées dans un format prédéfini et qui sont plus difficiles à stocker, rechercher et analyser que les données structurées.
- Les données non-structurées incluent des formats tels que le texte, les images, les vidéos, les fichiers audio, les fichiers PDF, les courriels, les réseaux sociaux, les documents Web, ….

III. Les principes de l'analyse sémantique des données d'entreprise.

- L'analyse sémantique des données d'entreprise est une méthode pour comprendre la signification des données stockées dans les systèmes d'information de l'entreprise.
- Elle implique l'utilisation de techniques telles que les ontologies, l'analyse de texte, l'apprentissage automatique et la collaboration avec les experts métier.
- Pourquoi l’analyse sémantique : pour maximiser la valeur des données d'entreprise (la valorisation des données, 5 ème V).

III. Les principes de l'analyse sémantique des données d'entreprise.

➔ Une ontologie est une description des concepts, attributs et des relations pouvant exister pour des données dans différentes colonnes. Par exemple:

- “client” est le concept
- “Date de naissance” et “nom” sont les attributs de ce concept.

L'utilisation d'ontologies peut aider à clarifier les relations entre les données et à faciliter la recherche et l'analyse de ces données.

III. Les principes de l'analyse sémantique des données d'entreprise.

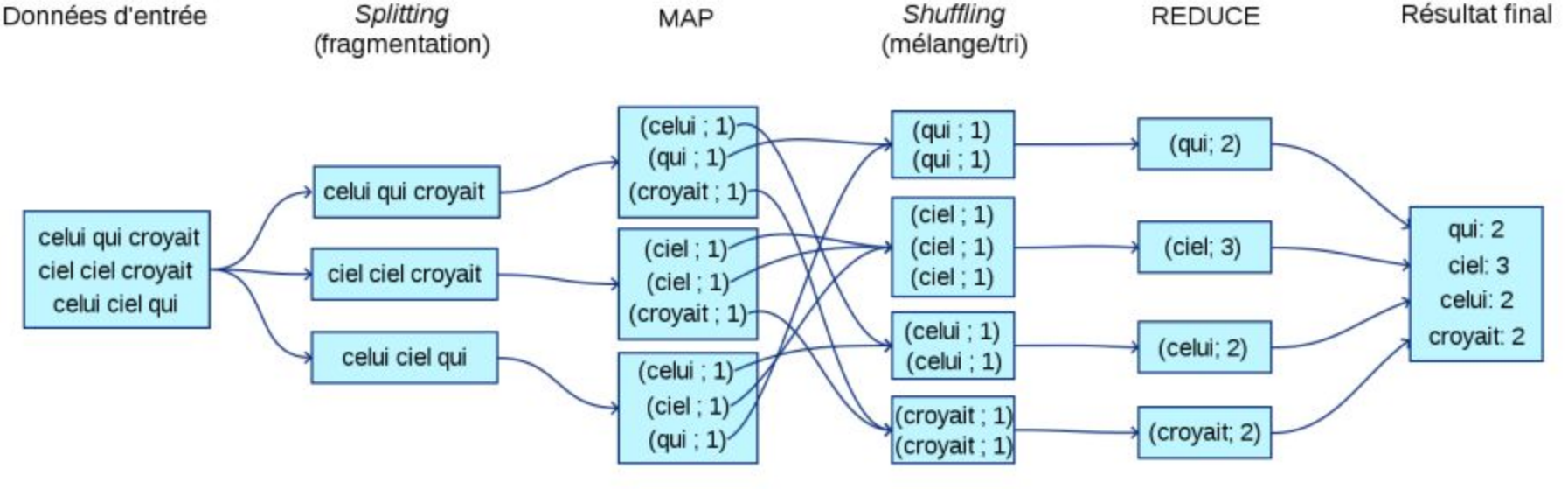
➔ L'analyse de texte est une méthode pour extraire des informations à partir de données textuelles, telles que des courriels, des documents Word, des pages Web, etc.

Les outils d'analyse de texte peuvent aider à extraire des informations précieuses à partir de ces sources de données

III. Les principes de l'analyse sémantique des données d'entreprise.

- L'apprentissage automatique est une méthode pour entraîner un algorithme à reconnaître des modèles dans les données et à en tirer des conclusions. Les techniques d'apprentissage automatique peuvent être utilisées pour aider à identifier des modèles cachés dans les données d'entreprise.

IV. Graphe des tâches à base de MapReduce.



Fin

Graphe des tâches à base de MapReduce

Chapitre III

Le stockage des données avec HBase

I. Plusieurs types de base de données XML

Il existe différents types de bases de données XML qui permettent de stocker et de manipuler des données au format XML :

1. Base de données XML natif : c'est une base de données qui est spécialement conçue pour stocker et manipuler des données XML. Les données sont stockées sous forme de documents XML dans la base de données, ce qui permet une manipulation facile des données

I. Plusieurs types de base de données XML

2. Base de données relationnelle avec support XML : certaines bases de données relationnelles telles que Oracle, IBM DB2 et Microsoft SQL Server incluent une prise en charge de XML.
3. Base de données graphe avec support XML : les bases de données de graphe, telles que Neo4j, incluent une prise en charge de XML. Cela permet de stocker des données XML dans la base de données de graphe et d'effectuer des requêtes sur ces données en utilisant des langages de requête tels que Cypher

II. Patterns d'usages et application au Cloud

- Les "Cloud patterns" (ou "modèles Cloud") sont des modèles de conception qui ont été spécialement développés pour résoudre des problèmes courants rencontrés dans le développement d'applications Cloud. Ces modèles sont basés sur des meilleures pratiques et des expériences éprouvées dans la conception d'applications pour les environnements Cloud.
- Les patterns Cloud sont souvent utilisés pour aider les développeurs à créer des applications qui sont adaptées aux environnements Cloud, qui peuvent être dimensionnées facilement, qui sont hautement disponibles, résilientes et qui utilisent efficacement les ressources de l'infrastructure Cloud

II. Patterns d'usages et application au Cloud

Ces modèles peuvent être appliqués à différents aspects de la conception d'une application, tels que l'architecture, la sécurité, la gestion des données, la gestion des ressources et la gestion des services. Exemples de patterns Cloud :

1. Autoscaling : permet à une application de s'adapter à la demande en augmentant ou diminuant automatiquement les ressources de calcul et de stockage en fonction de la charge de travail.

II. Patterns d'usages et application au Cloud

2. Load balancing : permet de distribuer le trafic réseau entre plusieurs serveurs pour répartir la charge de travail et garantir une disponibilité maximale de l'application.
3. Backups et récupération : permettent de sauvegarder les données de l'application et de les restaurer en cas de défaillance du système ou de catastrophe.
4. Microservices : permettent de créer des applications modulaires qui peuvent être facilement maintenues, mises à jour et dimensionnées en fonction des besoins

III. Le stockage des données HDFS & HBase

A. HDFS : Hadoop Distributed File System (HDFS)

B. HBase : HBase fait partie de la catégorie de SGBD orienté-colonne, et est l'un des premiers SGBD NoSQL à large échelle mis sur pied. Crée en fin d'année 2006 par Chad Walters et Jim Kellerman, HBase a rejoint l' écosystème Hadoop en octobre 2007 et en Juin 2010, il est devenu un projet prioritaire de la fondation Apache

A. Système de fichiers distribué

Qu’est-ce qu’un système de fichiers distribués ?

- Un système de fichiers distribué, ou DFS, est un système de stockage et de gestion des données qui permet aux utilisateurs ou aux applications d'accéder à des fichiers de données tels que des fichiers PDF, des documents Word, des images, des fichiers vidéo, des fichiers audio, etc., à partir d'un stockage partagé sur l'un des multiples serveurs en réseau. Avec des données partagées et stockées sur un cluster de serveurs, un DFS permet à de nombreux utilisateurs de partager des ressources de stockage et des fichiers de données sur de nombreuses machines.

A. Système de fichiers distribué

Les systèmes de fichier distribués les plus répandus sont :

- Ceph (https://docs.ceph.com/en/quincy/) est une plateforme libre de stockage distribué.
- GlusterFS (https://docs.gluster.org/en/latest/) est un système de fichiers réseau évolutif adapté aux tâches “data-intensive” telles que le stockage en nuage et le streaming multimédia.
- Google File System (GFS) est un système de fichiers distribué propriétaire. Il est développé par Google pour leurs propres applications
- Hadoop Distributed File System (HDFS) est un système de fichiers distribué, extensible et portable développé par Hadoop à partir du GoogleFS.

A. Système de fichiers distribué

Quand un système de fichiers distribués est-il essentiel ?

- Pour stocker des données de manière permanente.
- Pour partager facilement, efficacement et en toute sécurité des informations entre les utilisateurs et les applications.

A. Système de fichiers distribué

Avantages d'un système de fichiers distribués :

- Tolérance aux pannes
- Un accès local transparent
- Une indépendance par rapport à l'emplacement
- De capacités scale-out : pouvoir monter en charge massivement de façon parallèle en ajoutant plus de machines. Les systèmes DFS peuvent évoluer vers des clusters très importants qui comptent des milliers de serveurs.

B. HBase

HBase est un SGBD distribué, orienté-colonne qui fournit l'accès en temps réel aussi bien en lecture qu'en écriture aux données stockées sur le HDFS.HBase a été conçu pour :

- ne fonctionner que sur un cluster Hadoop ;
- être linéairement scalable, c'est-à-dire que supporte l'ajout de nœuds au cluster ;
- stocker de très grosses volumétries de données.
- fournir un accès en temps réel pour les opérations de lecture que d'écriture sur le HDFS ;
- s'appuyer sur des modèles de calculs distribués tels que le MapReduce pour l'exploitation de ses données ;

B. HBase

Une table HBase est un tableau multidimensionnel de données distribué et persisté sur le HDFS sous forme de fichiers spécifiques appelés HFiles. Voici les dimensions qui forment ce tableau multidimensionnel :

1. La première dimension c'est la « row key » (clé de ligne). Chaque ligne du tableau HBase est identifiée de façon unique par une clé « row key ». Attention, bien que similaire à une clé primaire dans une table relationnelle, la différence ici est que la row key n'est pas nécessairement une colonne parmi l'ensemble des colonnes désignée comme telle par le développeur, c'est une colonne interne à la structure de la table HBase, similaire à la colonne de numéros qui identifient chaque ligne d'une feuille de calcul Excel.

B. HBase

2. La deuxième dimension est la famille de colonne (column family), d'où HBase tire son nom de SGBD orienté-colonne. Les familles de colonnes représentent les valeurs d'un ensemble de colonnes physiquement colocataires c'est-à-dire physiquement sérialisées (stockées) dans le même fichier. Chaque famille de colonnes est persistée dans un HFile séparé, et chaque HFile possède son propre jeu de paramètres de configuration.

B. HBase

3. La troisième dimension est la colonne (column qualifier). Une colonne c’est l’adresse d’une série de données dans une famille de colonnes. Vous pouvez voir les colonnes HBase comme les colonnes d’un tableau ou comme les colonnes d’une feuille de calcul Excel. Ils désignent simplement les données des lignes de la même adresse dans une famille de colonne. Cette adresse porte un label et c’est ce label qu’on qualifie de colonne ou de qualificateur de colonne

B. HBase

Rqs:

- Les familles de colonnes sont statiques, c'est-à-dire définies à la création de la table et fixes tout au long de leur vie.
- Les colonnes sont dynamiques, c'est-à-dire ne sont pas définies lors de la création de la table, mais sont dynamiquement créées lors des opérations d' écriture/mise à jour de la donnée dans la colonne de la ligne.
- HBase traite les colonnes comme une table dynamique de paires de clé/valeur. Comme résultat, chaque ligne/famille de colonnes peut contenir des jeux arbitraires de colonnes.


clé: 1001
produit: livre Hadoop pour les consultants
Qtité: 7
Prix: 13 euros
clé: 1001
produit: livre Hadoop pour les consultants 2
Prix: 14 euros
clé: 1001
Qtité: 1
Prix: 19 euros
clé: 1002
produit: véhicule citadine
Qtité: 1
Prix: 13300 euros
clé: 1002
Prix: 13300 euros
clé: 1002
produit: véhicule citadine avec démarrage automatique
Prix: 16000 euros
clé: 1002
Qtité: 2
Prix: 1000 euros
clé: 1002
produit: costume 3 pièces marque LAMBROZINI
Prix: 1000 euros

B. HBase

4. La quatrième dimension c'est la cellule (cell) ou valeur (pour faire référence au contenu de la cellule). Une cellule HBase est l'intersection entre une row key, une famille de colonne et une colonne. Les données sont stockées dans les cellules.
5. Il faut savoir qu'HBase ne fait pas de différence entre les ajouts de lignes dans la table (opération d'écriture) et leur mise à jour (opération de modification d'une ligne). Tout comme dans le HDFS et dans un Data Warehouse, chaque opération de mise à jour est un ajout d'une nouvelle version de la même ligne de données. La distinction de versions d'une même ligne se fait à l'aide d'une valeur horodatée appelée TimeStamp.

B. HBase

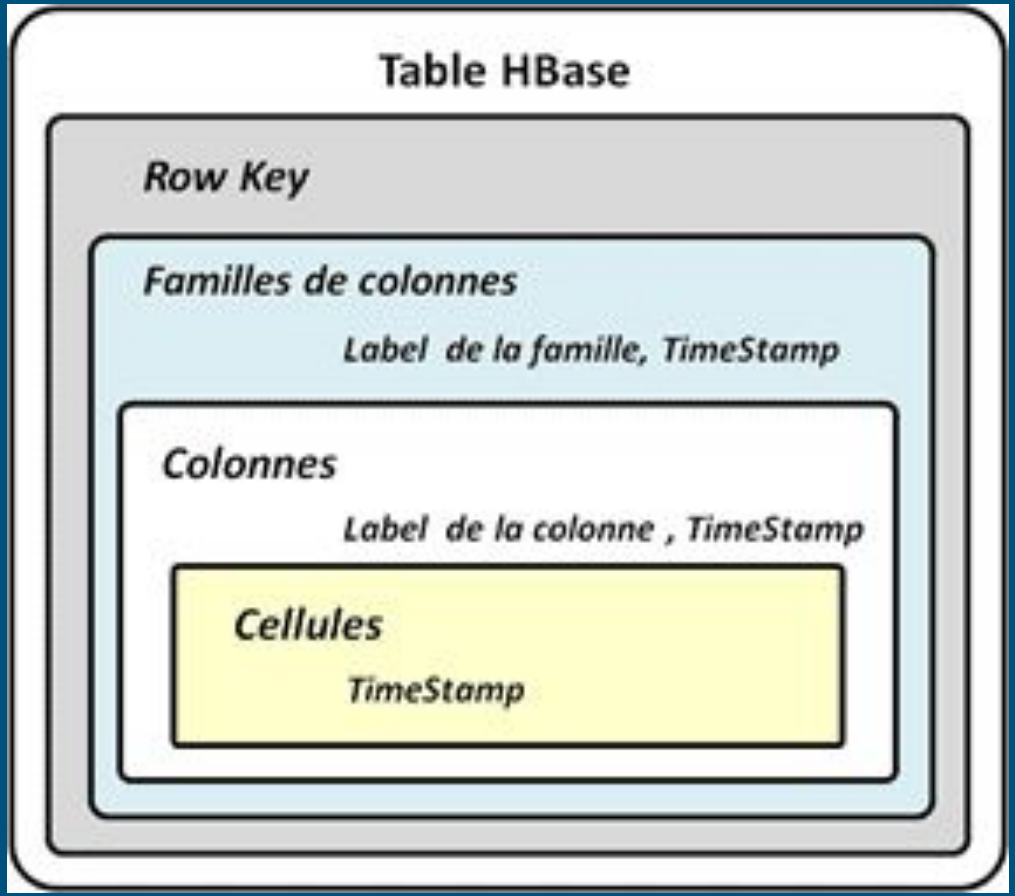
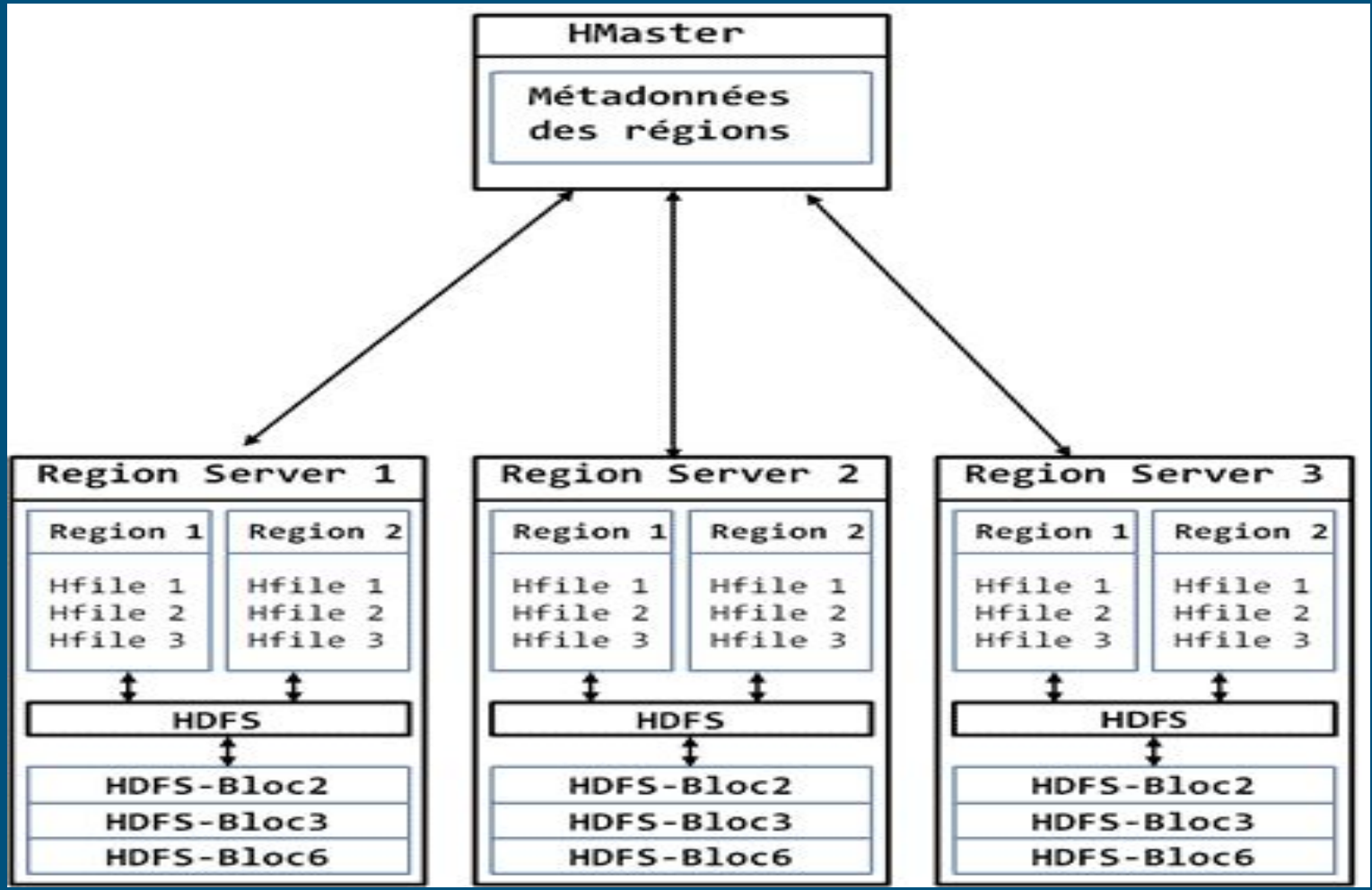


Table HBase
Row Key
Familles de colonnes
Label de la famille, TimeStamp
Colonnes
Label de la colonne , TimeStamp
Cellules
TimeStamp

B. Architecture d’un cluster HBase

- Le HBase utilise le HDFS pour stocker les tables.
- Le HDFS voit HBase comme un client à qui il fournit du support de stockage des HFiles qui constituent les tables en blocs.
- Les tables HBase sont persistées sur le disque sous forme de fichiers HDFS appelés HFiles. Chaque HFile contient les données d’une et une seule famille de colonnes.
- Toutes les lignes de la table sont identifiées de façon unique à l’aide d’une valeur de la row key.


HMaster
Métadonnées des régions
Region Server 1
Region 1
Hfile 1
Hfile 2
Hfile 3
Region 2
Hfile 1
Hfile 2
Hfile 3
HDFS
HDFS-Bloc2
HDFS-Bloc3
HDFS-Bloc6
Region Server 2
Region 1
Hfile 1
Hfile 2
Hfile 3
Region 2
Hfile 1
Hfile 2
Hfile 3
HDFS
HDFS-Bloc2
HDFS-Bloc3
HDFS-Bloc6
Region Server 3
Region 1
Hfile 1
Hfile 2
Hfile 3
Region 2
Hfile 1
Hfile 2
Hfile 3
HDFS
HDFS-Bloc2
HDFS-Bloc3
HDFS-Bloc6

B. Architecture d’un cluster HBase

- Etant donné que la table fait office de base de données, pour une application métier donnée, toutes les données sont stockées dans la seule table HBase, qui pourra alors rapidement contenir des milliards de lignes, chiffrant sa taille en Téra octets voir Péta-octets.
- Cette taille phénoménale rend impossible le stockage de la table sur une seule machine.
- Pour résoudre ce problème, les tables HBase sont divisées en partitions appelées « régions » qui sont réparties entre les nœuds RegionsServers pour le stockage.

B. Architecture d’un cluster HBase

- Le HBase utilise le HDFS pour stocker les tables.
- Le HDFS voit Les régions sont distribuées sur le cluster de façon aléatoire et chaque nœud RegionsServer peut stocker une ou plusieurs régions.
- Les régions sont répliquées entre les nœuds de façon à maintenir la disponibilité du cluster en cas de panne.

Fin