Management et déploiement de solutions Big Data

Animé par Sylvain Labasse

2 500 000 000 000 000 000

EN FIN DE MODULE, VOUS SAUREZ…

Organiser un cluster Hadoop

Concevoir un traitement MapReduce

Coder un mapper et un reducer en java

Structurer une base de données orientée colonne

Interroger une base orientée colonne

Pre-Requis

Public

I5

Nécessaire

Infra : Linux, Shell script

Dev : Linux, Java

Support

Copie des slides sur MyLearningBox

Notes de cours

Réalisation des ateliers

L'environnement

Matériel

Ordinateur connecté sous Windows, Linux ou MacOS

Logiciel

VirtualBox, VM Hadoop (distribution cloudera)

EVALUATION

Ateliers

A remettre sur MyLearningBox en fin de séance

QCM (individuel)

Contenu

Le BigData

- NoSQL, Big Data
- Hadoop
- Mise en œuvre

HDFS

- Vue d'ensemble
- Clusters et services
- Administration
- Usage console et Java

Contenu (suite et fin)

MapReduce

- Principe
- Implémentation en Java
- Hadoop Streaming

HBase

- Structure
- Administration
- HBase shell
- MapReduce

LE BIG DATA

Objectifs

Enjeux du NoSQL et du BigData

Panorama des solutions

Intérêt des distributions Hadoop

Le Big Data

- ➔ NoSQL
- Big Data
- Hadoop
- Mise en œuvre
- Résumé

NoSQL – Origines

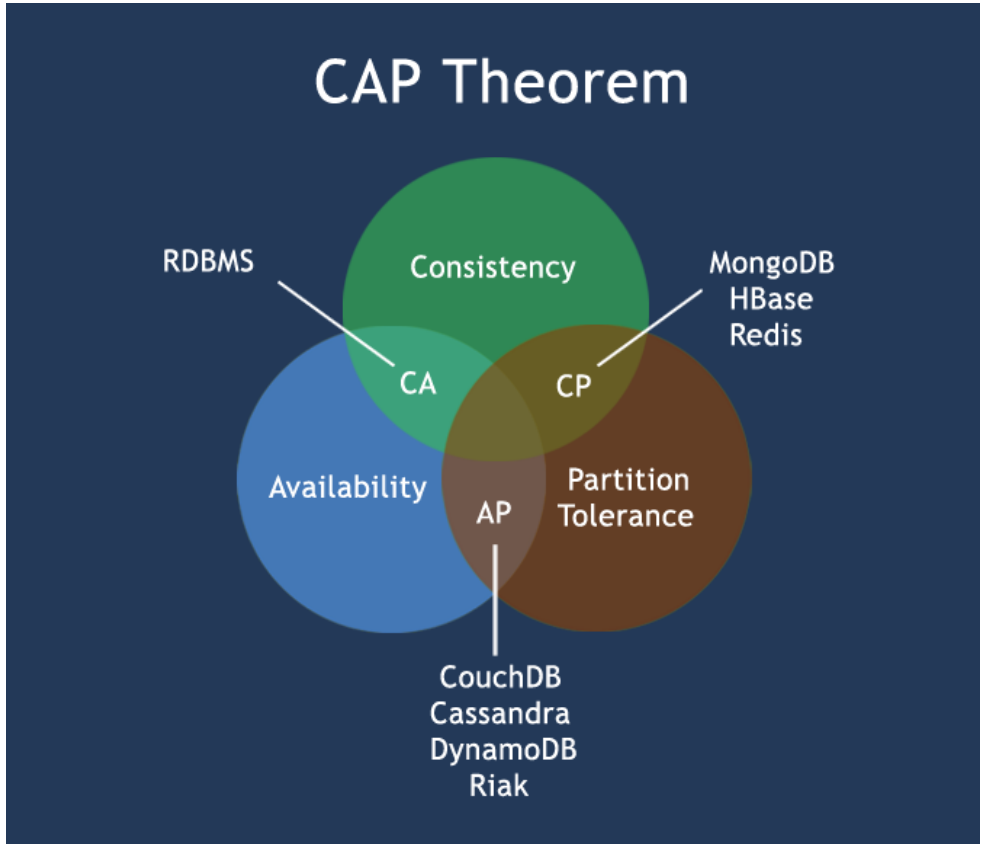

Figure 1 - Théorème CAP (http://www.abramsimon.com)

NoSQL

Not Only SQL

Pas forcément ACID[1]

Avantages : Montée en charge, Structure plus libre

Plusieurs types

Colonne avec notion de Column family

Document

Clé/Valeur

Graphe

[1] Atomicité, Consistance, Isolation, Durabilité

NoSQL – Panorama

Colonne

Cassandra, **HBase**, SAP HANA

Document

CouchDB, MongoDB

Clé-valeur (Key-value)

Redis, Dynamo, Riak, Oracle NoSQL Database

Graphe

Neo4J

NoSQL – Matrice de choix

	Performance	Montée en charge	Flexibilité	Complexité	Fonctionnalités
SGBDR	Variable	Variable	Faible	Modérée	Algèbre relationnel
Colonne	Haute	Haute	Modérée	Basse	-
Document	Haute	~Haute	Haute		-
Clé-valeur	Haute	Haute	Haute	Aucune	-
Graphe	Variable	Variable	Haute	Haute	Théorie des graphes

Le Big Data

- ✓ NoSQL
- ➔ Big Data
- Hadoop
- Mise en œuvre
- Résumé

Big Data

3V

Volume : Trop pour être gérées sur 1 machine
Variété : Sources et formats divers
Vitesse : Parallélisation nécessaire

5V

Véracité : Evaluation/Elimination données incohérentes
Valeur : Déductions/Anticipation à partir de l'existant

Le Big Data

- ✓ NoSQL
- ✓ Big Data
- ➔ Hadoop
- Mise en œuvre
- Résumé

HADOOP

Historique

2003 – Google : Livre blanc GFS, 2004 : article sur MapReduce

2004 – Portage par Doug Cutting (Yahoo!) pour Apache

2008 – Hadoop est top level project au sein d'Apache

Framework

Cœur : HDFS + MapReduce + Yarn (resources, monitoring)

Bdd : HBase, Cassandra / Exploit. : Hive, Spark, Mahout (ML)

Gestion : Zookeeper, Ambari (web), distcp, Ooozie (planif)

ETL : Spark, Sqoop (SGBDR), Avro (serial.), Kafka/Storm (flux)

Le Big Data

- ✓ NoSQL
- ✓ Big Data
- ✓ Hadoop
- ➔ Mise en œuvre
- Résumé

Mise en œuvre

Difficultés

Besoin d'homogénéité des clusters

Incompatibilités d'outils entre versions

Distributions

Cloudera : Noyau OpenSource + propriétaire

HortonWorks : Racheté par Cloudera

MapR : OpenSource+Payant, MapR FS (offre Amazon)

Pivotal : Des outils propriétaire (orienté Analytics)

Le Big Data

- ✓ NoSQL
- ✓ Big Data
- ✓ Hadoop
- ✓ Mise en œuvre
- ➔ Résumé

Resume

Enjeux du NoSQL et du BigData

Panorama des solutions

Intérêt des distributions Hadoop

HDFS

Objectifs

Fondements d'un système réparti de fichiers

Rôles des services HDFS

Gestion des clusters et de la haute dispo.

Manipulation en console ou java

HDFS

- ➔ Vue d’ensemble
- Clusters et services
- Administration
- Usage console et Java
- Résumé

VUE D'ENSEMBLE

Hadoop Distributed File System

Distribué

Répliqué : ratio à définir (défaut : 3)

Par bloc (défaut : 64Mo)

Optimisé pour Write once, read multiple (pas de modif)

HadoopFS

Couche d'abstraction d'accès à HDFS

Accès par URI : hdfs://, file://, ftp://, s3://, gfs://, …

HDFS

- ✓ Vue d’ensemble
- ➔ Clusters et services
- Administration
- Usage console et Java
- Résumé

HDFS – Clusters

Stockage et compression

Répertoire sur l'index

Fichier sur le cluster

Au choix, transparent pour l'appli

Conseil : gzip/bzip lecture, lzo/snappy pour transit.

Permissions

Type linux : user, group, sticky sur répertoire

Possibilité d'ACL posix

Désactivable : Super User par login/mdp ou kerberos

HDFS – Services

NameNode (1)

- Hiérarchie des répertoires
- Index des fichiers et répartition des blocs (FSImage)
- EditLog (journal des modifs)

DataNode (1..*)

- Données des fichiers
- Eviter RAID

HDFS – Lecture/Ecriture

Lecture

Demande au NameNode → Liste des blocs/DataNode

Demande au DataNode → Blocs à assembler

Ecriture

Taille au NameNode → Réservation blocs/DataNode

Ecriture sur DataNode →DataNode prévient NameNode

Sous-réplication → NameNode demande la réplication

HDFS

- ✓ Vue d’ensemble
- ✓ Clusters et services
- ➔ Administration
- Usage console et Java
- Résumé

ADMINISTRATION

Fsck

Vérification intégrité, corrections éventuelles (move/del)

> hdfs fsck /

Balancer

Equilibre le taux de remplissage à +/- 10%

Opération longue : plusieurs heures/jours

> sudo hdfs balancer

Administration (suite)

Fédération de clusters

Isolation d'arborescence

Cycle de vie

Snapshot : Versionning du FS

Corbeille : Activée par défaut

Journal Node

HA par Quorum : Nb impair, perte possible de $\frac{n-1}{2}$ nœuds

NameNode actif et NameNodes en standby+safemode

HDFS

- ✓ Vue d'ensemble
- ✓ Clusters et services
- ✓ Administration
- ➔ Usage console et Java
- Résumé

En console

hdfs dfs -cmd

Avec l'utilisateur courant (répertoire courant = home)

sudo -u pour impersonnifier (Superutilisateur = hdfs)

help, ls, mkdir, ~~cd~~, …

Transfert hdfs/local

put fichier_local chemin_hdfs

getmerge répertoire_hdfs : équiv. cat *

get fichier : Récupère le fichier

text fichier.gz : Texte contenu dans l'archive

En Java

A savoir

Package : org.apache.hadoop, org.apache.hadoop.fs

Objets : FileSystem, Path

Exemple

```
Configuration conf = new Configuration();
try {
    URI uri = new URI("hdfs://url_namenode");
    FileSystem fs = FileSystem.get(uri, conf);
    Path root = new Path(uri);

    System.out.println(fs.getFileStatus(new Path(root, "/chemin/absolu/fichier")));
} catch (URISyntaxException    e) { e.printStackTrace();
} catch (FileNotFoundException e) { e.printStackTrace();
} catch (IOException           e) { e.printStackTrace();
}
```

HDFS

- ✓ Vue d’ensemble
- ✓ Clusters et services
- ✓ Administration
- ✓ Usage console et Java
- ➔ Résumé

Resume

Fondements d'un système réparti de fichiers

Rôles des services HDFS

Gestion des clusters et de la haute dispo.

Manipulation en console ou java

MAP REDUCE

Objectifs

Parallélisation grâce au paradigme MapReduce

Rôles du mapper et du reducer

Ecriture d'un map reduce en java

Lancement par Hadoop Streaming

MAP REDUCE

➔ Principe

Implémentation en Java

Hadoop Streaming

Résumé

Principe

Etapes

Split : Répartition des données sur chaque nœud

Map : Production de couples (clé, valeur)

Shuffle and sort : Regroupement par clé

Reduce : Traitement de toutes les valeurs pour 1 clé

Exemple

Casting de films

Fonction Reduce

Contraintes

- Gestion stricte de la mémoire
- Stateless

Liste de valeurs

- Liste de taille fixe
- Ou Itérateur

MAP REDUCE

- ✓ Principe
- ➔ Implémentation en Java
- Hadoop Streaming
- Résumé

Mapper - Java

extends Mapper<KeyIn, ValIn, KeyOut, ValOut>

Tous les types sont Writable

Substitution map

Writable

Byte/Short/Int/Long/VInt/VLongWritable, NullWritable

BytesWritable, Text, MD5Hash, Record

Map

```
public void map(KeyIn keyIn, ValIn valIn) {
    /*…*/ context.write(keyOut, valOut) ; /*…*/
```

Reducer – Java

extends Reducer<KeyIn, ValIn, KeyOut, ValOut>

KeyIn,ValIn du reducer = KeyOut, ValOut du mapper

Substitution de reduce

Reduce

```
public void reduce(KeyIn keyIn, Iterable<ValIn> valsIn) {
    //…
    context.write(keyOut, valOut) ;
}
```

Driver – Java

extends Configured implements Tool

Implémentation run et main

main()

```
Configuration conf = new Configuration();
ToolRunner.run(conf, new XxxDriver(), args));
```

run()

Paramétrage : jar, mapper, reducer, nb tâches reduce

Définition des formats d'entrée/sortie

```
return job.waitForCompletion(true) ? 0 : -1;
```

Driver Java – run()

Paramétrage

job = Job.getInstance(this.getConf(), "jobName");
job.setJarByClass : Envoi sur nœuds
setMapperClass, setReducerClass
job.setNumReduceTasks(10) ; // optionnel

Format entrées/sorties

job.setMapOutputKeyClass, job.setMapOutputValueClass
job.setInputFormatClass(TextInputFormat.class);
FileInputFormat.addInputPath(job, new Path("/in"));
job.setOutputFormatClass(TextOutputFormat.class);
FileOutputFormat.setOutputPath(job, new Path("/out"));

Lancement

Ligne de commande

hadoop jar xxx.jar -D mapred.reduce.tasks=10

Affichage d'un lien de suivi du job

Résultat

_SUCCESS

+ 1 fichier / tâche reducer

Getmerge

Map Reduce

- ✓ Principe
- ✓ Implémentation en Java
- ➔ Hadoop Streaming
- Résumé

Hadoop Streaming

Intérêt

Mapper/reducer utilisent entrées/sorties standard
Utilisation de n'importe quel langage/script/commande

Ligne de commande

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar
-input /path/to/input –output /path/to/output
-mapper scriptmapper -reducer scriptreducer
-files scriptmapper scriptreducer
-numReduceTasks 10
```

MAP REDUCE

- ✓ Principe
- ✓ Implémentation en Java
- ✓ Hadoop Streaming
- ➔ Résumé

Resume

Parallélisation grâce au paradigme MapReduce

Rôles du mapper et du reducer

Ecriture d'un map reduce en java

Lancement par Hadoop Streaming

HBASE

Objectifs

Organisation de HBase

Administration et supervision d'un cluster HBase

Interrogation d'un HBase : Batch ou MapReduce

HBase

➔ Structure

Administration

HBase Shell

MapReduce

Résumé

Structure

Table

- Clé de ligne
- Ensemble de familles de colonnes

Colonne

- Appartient à une famille de colonnes
- Possède un identifiant de colonne

Cellule

- Ligne / Colonne
- Versionné

Structure – Stockage

Région

Intervalle de clés

1 région = 1 Serveur

Entrées

Clé

Colonne = famille:nom

Version

Valeur

Operations

Définition

Create, Describe, Alter, Drop

Enable, Disable

Manipulation

Get : Interroge 1 ligne ou 1 cellule

Delete : Supprime 1 ligne ou 1 cellule

Put : Ecrit une ligne (id ligne, valeurs pour colonnes)

Scan : Parcours les lignes

HBase

- ✓ Structure
- ➔ Administration
- HBase Shell
- MapReduce
- Résumé

Administration

Services

HBase master

HBase regionserver (plusieurs)

Zookeeper : état, assignation des régions

HA

HBase master multiples (2 – +∞)

Min 3 zookeepers par Quorum

Impair, perte possible de $\frac{n-1}{2}$ nœuds

Zookeeper répertorie regionservers actifs

ADMINISTRATION – OPERATIONS

Interfaces

- HBase REST
- HBase Thrift

Opérations

- Equilibrage des régions
- Haute disponibilité

HBase

- ✓ Structure
- ✓ Administration
- ➔ HBase shell
- MapReduce
- Résumé

Commande HBase Shell

Création / remplissage

> create 'nomtable', 'famille1', { NAME => 'f2', *options*}

Options : compression, versions, …

> put 'nomtable', 'clé', 'famille1:champ1', 'valeur'

« put » insère ET modifie sur la même clé

Scans et filtres

> scan 'nomtable', { *options* }

Résultat : columns, versions, reversed, limit, …

Filtre : FILTER => org.apache.hadoop.hbase.filter.KeyOnlyFilter.new('clé')

HBase

- ✓ Structure
- ✓ Administration
- ✓ Scans et filtres
- ➔ MapReduce
- Résumé

MapReduce

TableMapper

Entrées : ImmutableBytesWritable+Result

Conversions Bytes : Bytes.toString / Bytes.toBytes

TableReducer

Produit une ou plusieurs opérations HBase

Sortie : NullWritable, Mutation (new Put/Delete)

Driver

TableMapReduceUtil.initTableMapJob

TableMapReduceUtil.initTableReduceJob

HBase

- ✓ Structure
- ✓ Administration
- ✓ HBase Shell
- ✓ MapReduce
- ➔ Résumé

Resume

Organisation de HBase

Administration et supervision d'un cluster HBase

Interrogation d'un HBase : Batch ou MapReduce

BILAN

MAINTENANT, VOUS POUVEZ…

Organiser un cluster Hadoop

Concevoir un traitement MapReduce

Coder un mapper et un reducer en java

Structurer une base de données orientée colonne

Interroger une base orientée colonne