Les bases de données relationnelles (SQL) atteignent actuellement leurs **limites** de **performances** et de **stockage** mais un nouveau modèle, le NoSQL (Not Only SQL) permet désormais de franchir ces limites.

Ces bases sont beaucoup **moins couteuses** car elles peuvent fonctionner sur du **matériel peu cher**. Elles **s'adaptent** également **facilement**. Cela vient du fait qu'il n'y a pas de limites sur le nombre de disques dur de la base de données, et que le système voit le système de stockage comme une seule entité. Les disques sont donc faciles à ajouter ou à enlever, et n'importe quel disque peut être ajouté, notamment des disques peu couteux.

Nous nous intéresserons dans ce document au framework Hadoop, qui est l'éléphant du BigData et est utilisé par la majorité des utilisateur de BigData.

Hadoop est **un Framework Java libre** permettant de développer des applications distribués et échelonables (scalables). Il est utilisé pour les bases de données et regroupe de nombreuses librairies.

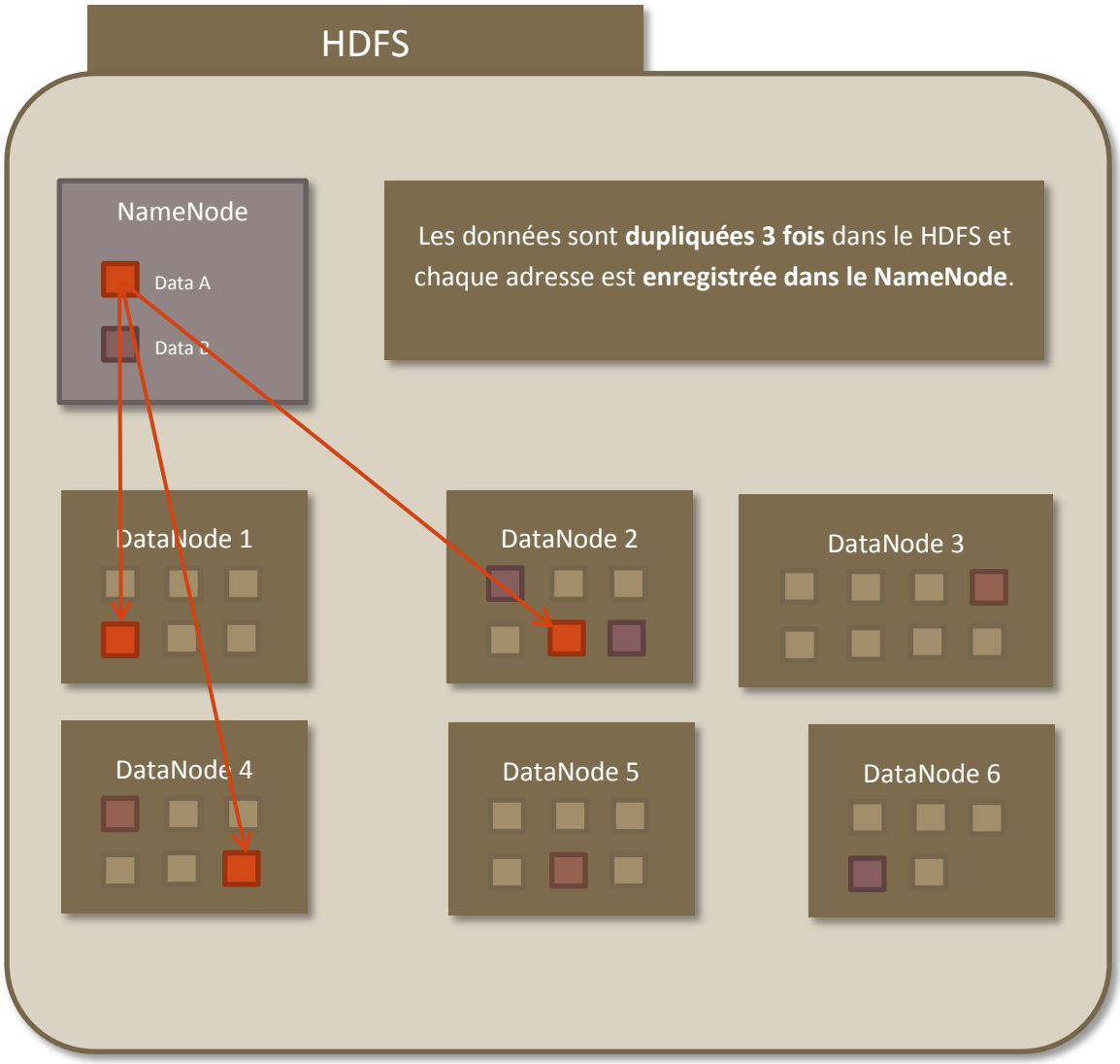
Hadoop est basé sur le **HDFS** (Hadoop Distributed File System) qui est un modèle de stockage et sur l'algorithme **MapReduce** qui permet d'effectuer des calculs parallèles et d'avoir des données fiables.

Nous nous intéresserons dans un premier temps aux indispensables d'Hadoop et nous verrons ensuite les outils utilisés dans une architecture classique d'Hadoop.

TABLE DES MATIERES

LES INDISPENSABLES D'HADOOP : HDFS ET MAPREDUCE

I. LE NAMENODE



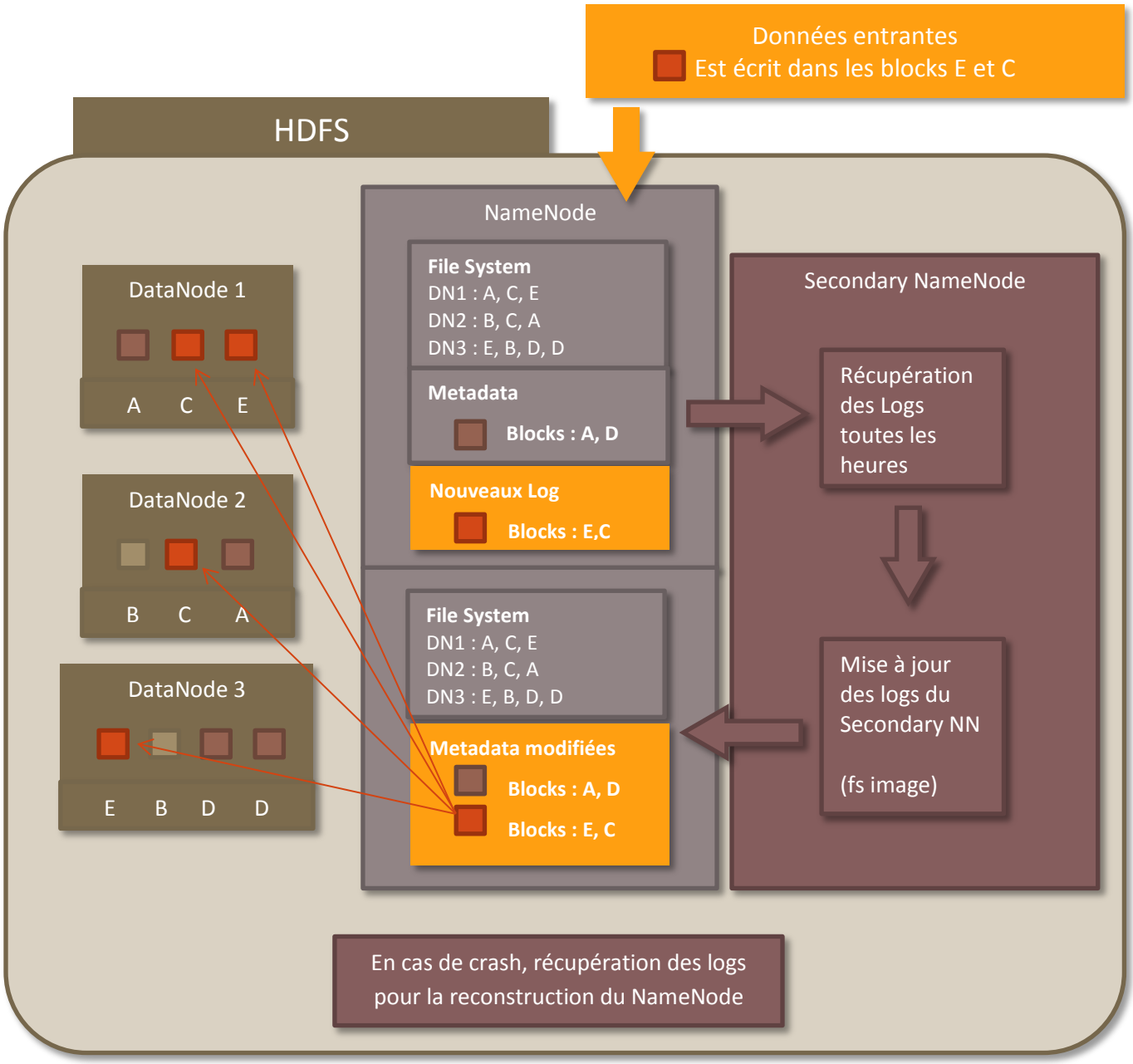
Chaque disque dur est un **nœud** (certains outils permettent tout de même d'avoir plusieurs disques sur un node ou plusieurs node sur un disque : ref Sheepdog). Les disques peuvent être plus ou moins grand.

Les données sont répliquées trois fois afin d'assurer leurs **validité**. Elles peuvent être répliquées sur des nœuds différents et elle peuvent également être dupliquées plus ou moins de 3 fois.

L'avantage d'avoir un système distribué est que l'ajout de disque est facile. Les disques peuvent être **peu cher**, pas besoin de tout stocker sur un même disque ou de réadapter l'application à chaque fois que les disques sont trop chargés.

L'autre avantage est que les processus s'exécutent sur plusieurs disque. Les **performances** sont donc grandement **augmentées**. On a une parallélisation massive de

II. LES SECONDARY NODE, BACKUP NODE ET CHECKPOINT NODE

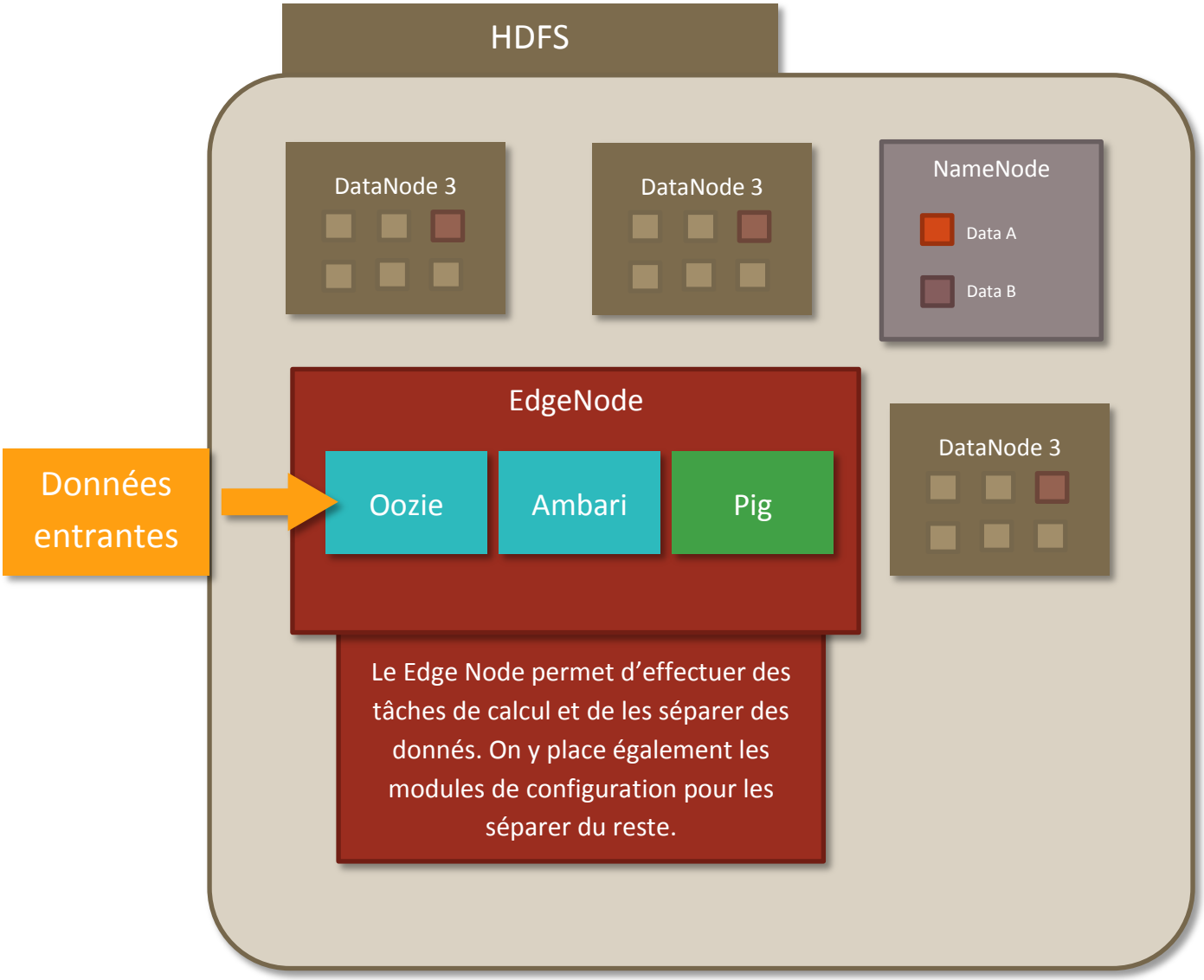


Le NameNode enregistre les modifications des fichiers sous forme de Logs et le secondary NameNode les récupères pour ensuite modifier le NameNode. En cas d'arrêt du NameNode tous les logs reçus auront été traités par le NameNode, ce qui permet de récupérer rapidement un état stable.

Hadoop reste cependant un **SPOF** (Single Point Of Failure) et le **secondary NameNode est surtout là pour répartir la charge sur le namenode** et aussi l'aider en cas de crash. Il ne prend cependant **pas le relai du NameNode** en cas de panne mais lui permet de ne pas perdre de données et de lui faire récupérer son état rapidement. Il existe cependant des solutions pour remédier à ce SPOF :

D'autres nodes existent pour aider le NameNode : le checkpoint Node qui copie le NameNode en temps-réel et met à jour ses logs directement. Le Back-up Node est probablement le futur de ces Node et offre les même fonctionnalités que le CheckPoint Node et permet au NameNode de lire dans ses données n'importe quand.

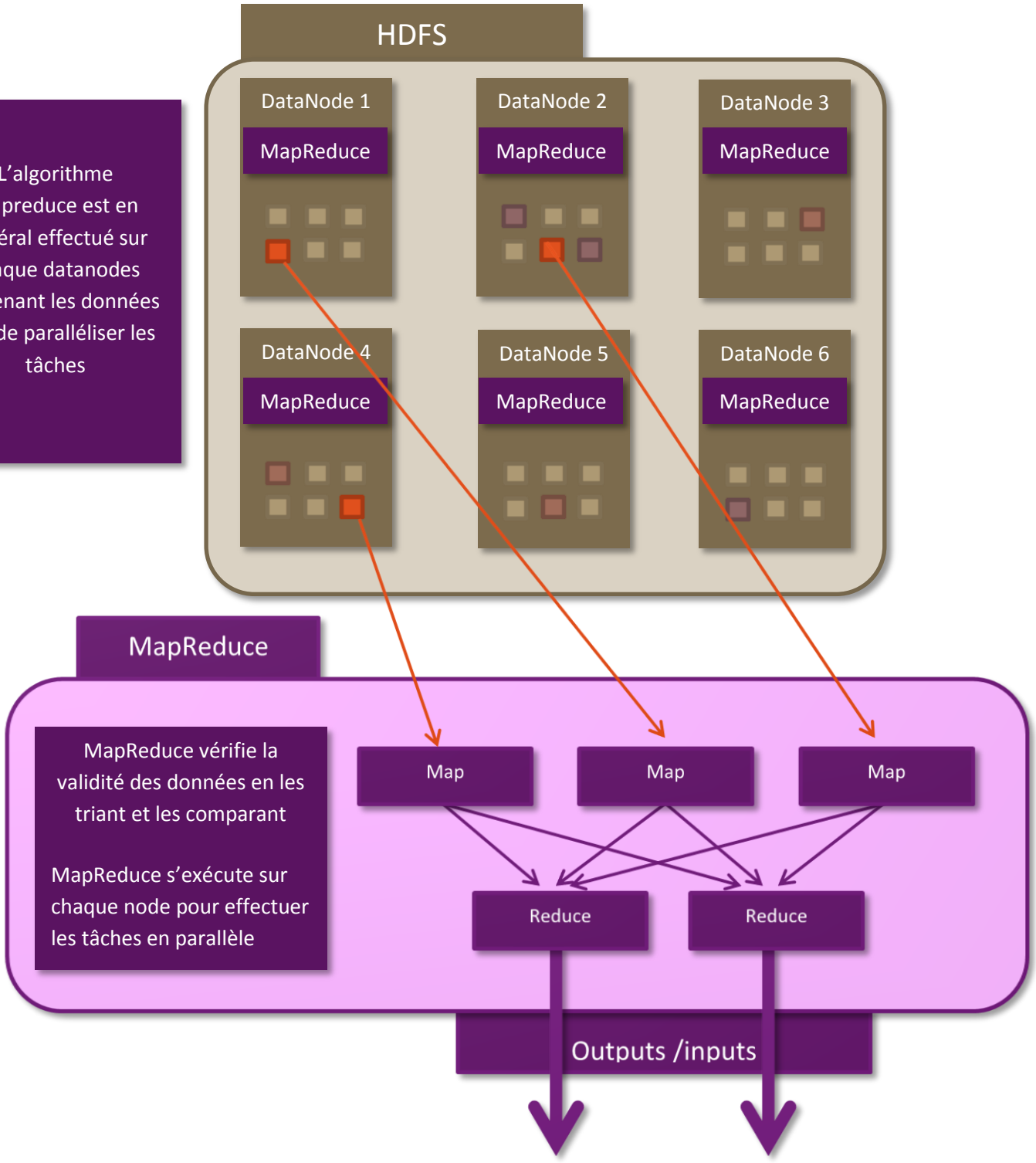
III. LE EDGENODE



Le Edge node est généralement utilisé en entrée de la base mais sert aussi à des traitements nécessitant beaucoup de calculs et aux modules d'administration. Cela permet que les **NameNodes** ne soient **pas sollicités pour des tâches autres que la recherche dans ses données**, et de cibler le matériel pour des traitements spécifiques.

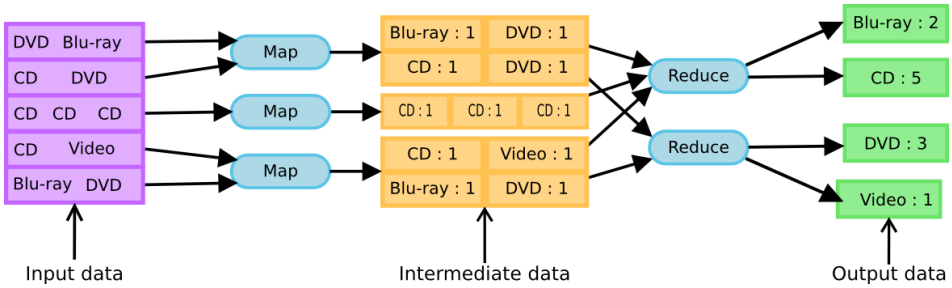
IV. MAPREDUCE

L'algorithme mapreduce est en général effectué sur chaque datanodes contenant les données afin de paralléliser les tâches



MapReduce permet d'organiser les données à l'enregistrement, et éventuellement de les trier

VOIR Hive page 11



Fonctionnement de MapReduce

V. FILE / COMPRESSION

Les fichiers Hadoop peuvent avoir n'importe quel forme, les valeurs étant enregistrées en binaires. On peut cependant les compresser :

Compression	File	Size(GB)	Compression Time(s)	Decompression Time(s)
None	logs	8.0	-	-
Gzip	logs.gz	1.3	241	72
LZO	logs.lzo	2.0	55	35
Snappy	-	4.2	40	27

Hadoop est plus performant et moins gourmand en mémoire pour les gros fichiers et pour des données ayant des milliers/millions de colonnes différentes.

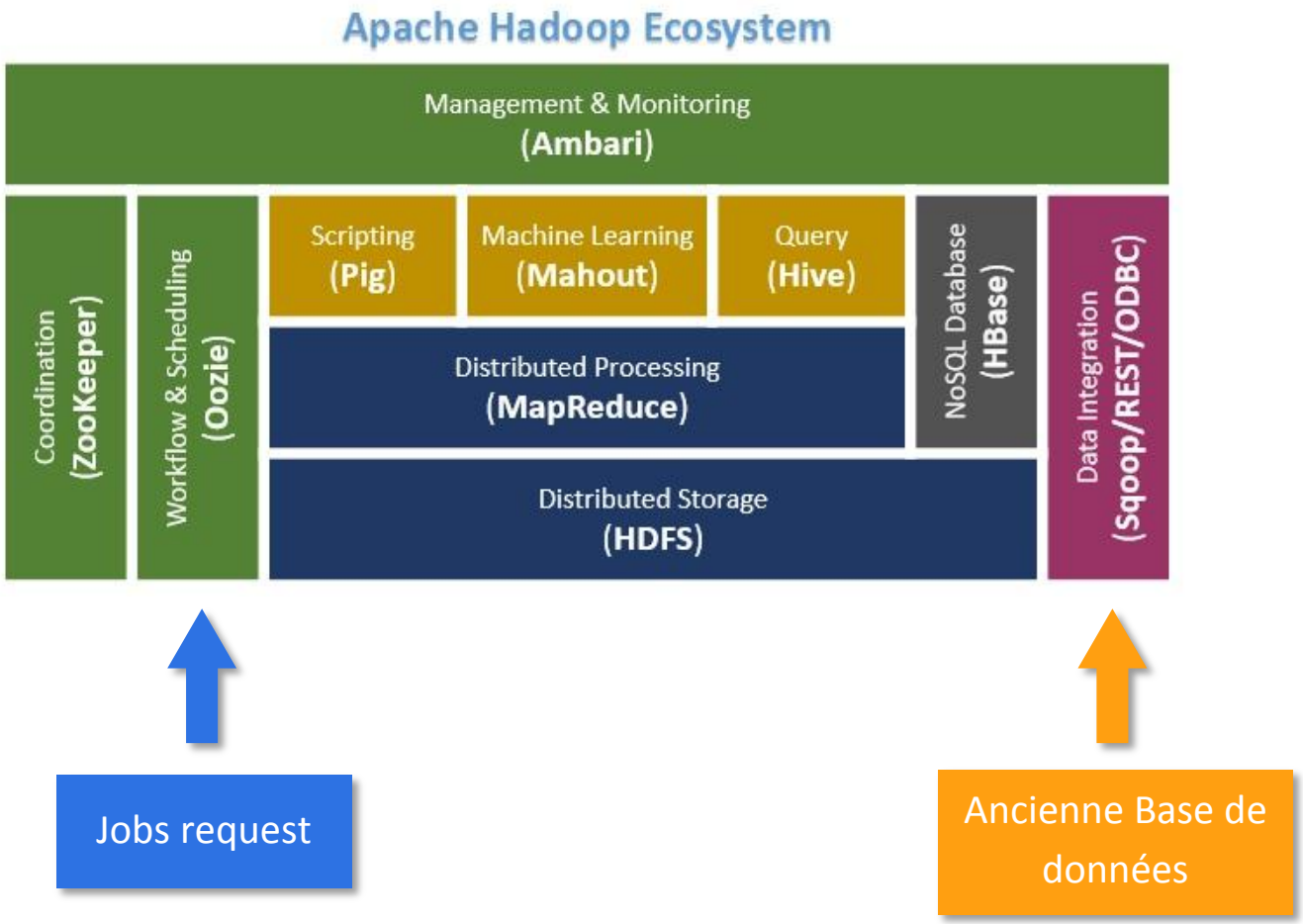
L'alternative a Hadoop orienté écriture et pour des fichiers plus petits est Cassandra qui se rapproche beaucoup d'une base de données relationnelle.

Les données peuvent être compressées en Gzip, LZO, Snappy. D'après les documents vus (ne pas hésiter à me contacter si je me trompe), on accède avec Hive (~query access) plus rapidement si les données sont compressées que si elles ne le sont pas.

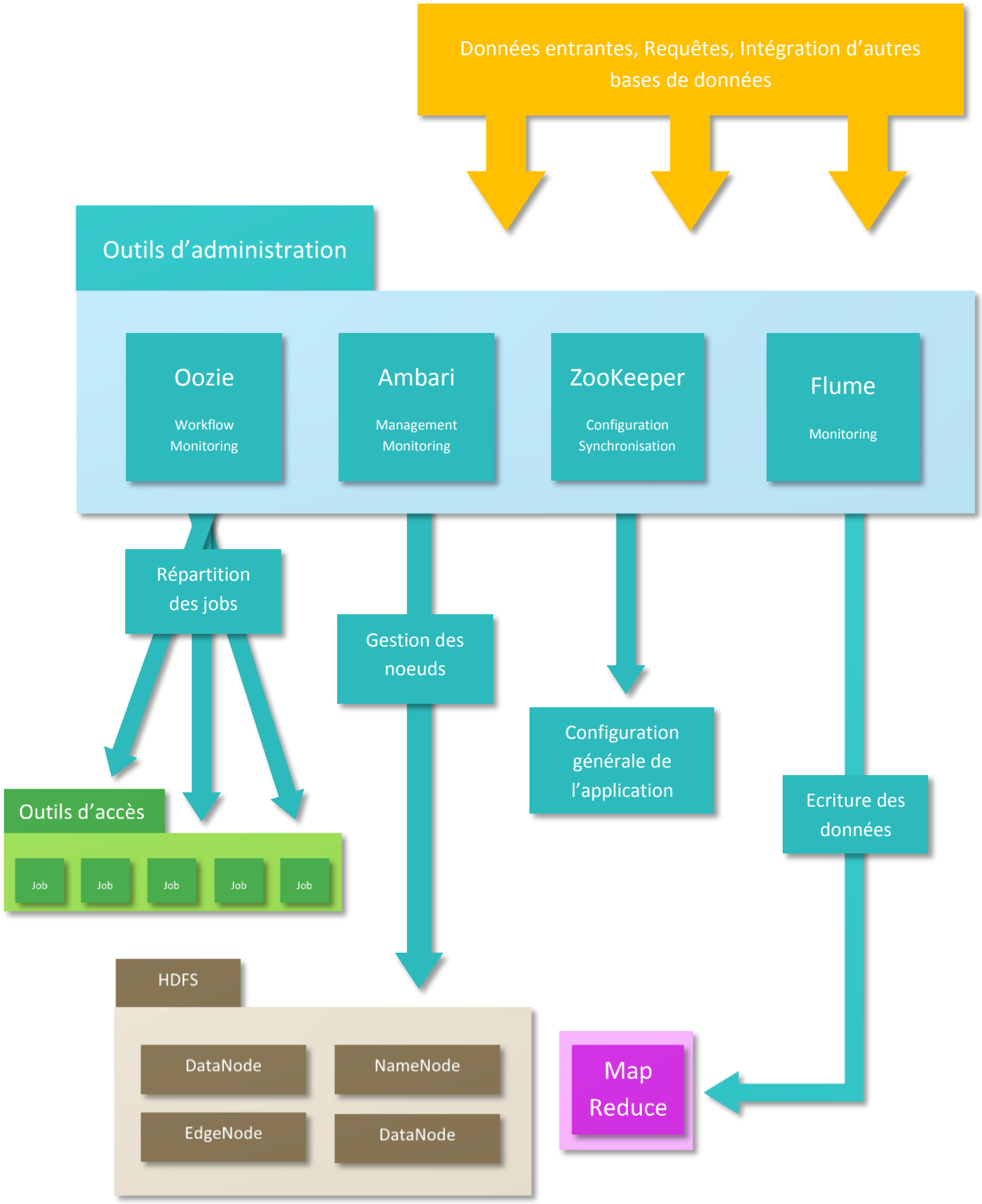
LZO or Snappy – fast but sloppy – Best for temporary tables – ZLIB – slow and complete – Best for long term storage

ECOSYSTEME HADOOP

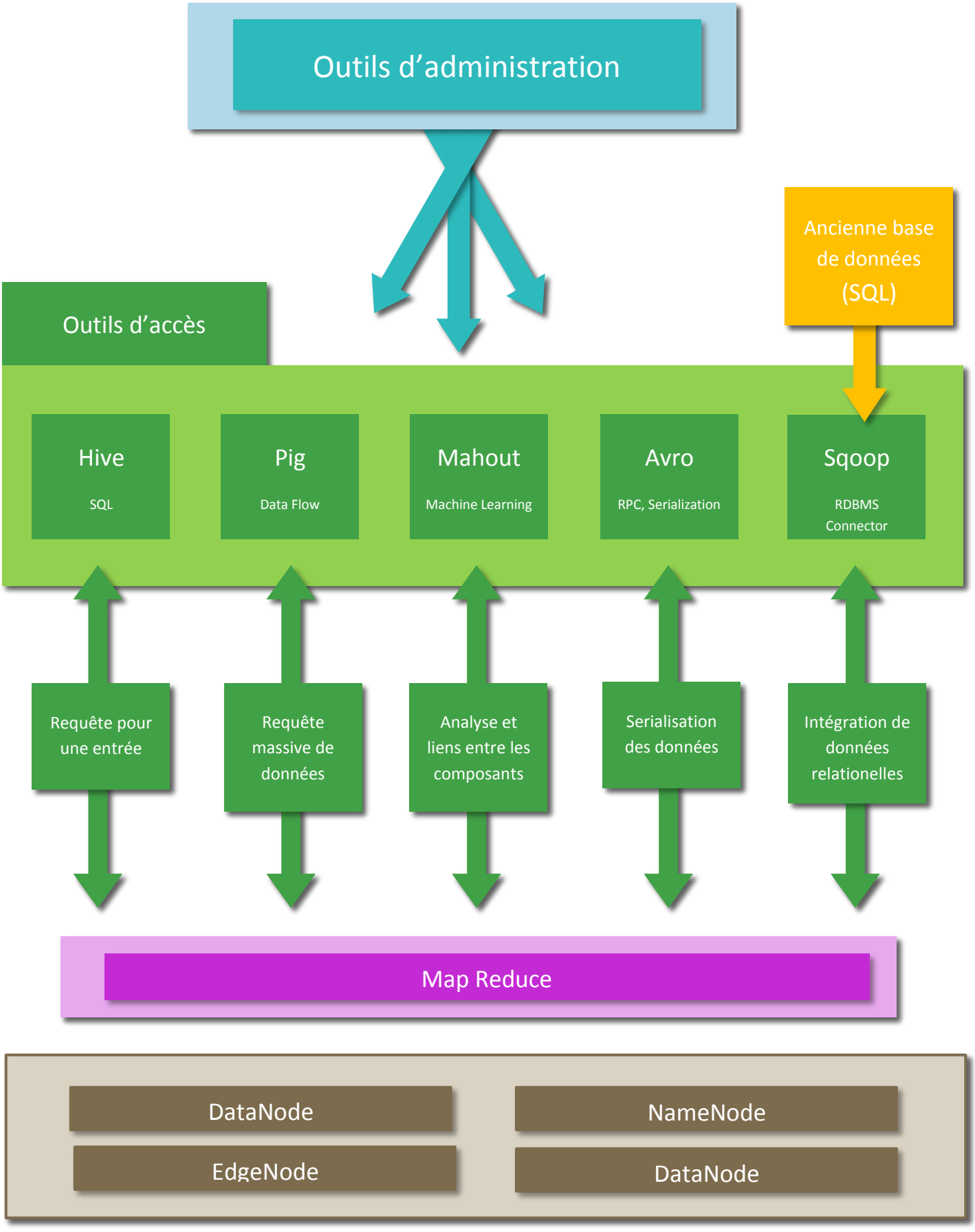
L'écosystème Hadoop est gigantesque et de nombreux outils sont utiles à des solutions spécifiques. Ceux présentés ci-dessous sont les principaux.



VI. OUTILS DE GESTION ET CONFIGURATION DE L'APPLICATION



VII. OUTILS D'ACCES AUX DONNEES



IX. OUTILS

1. HIVE

L'avantage d'Hive est qu'il permet le partitionnement des données par colonnes et d'y accéder facilement. Un bref exemple :

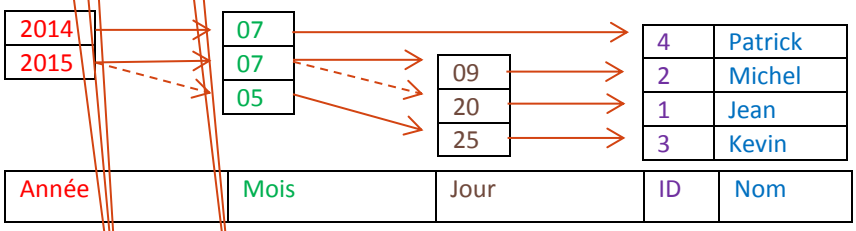
SQL

Année	Mois	Jour	ID	Nom
2014	07	Vide	4	Patrick
2015	07	09	2	Michel
2015	07	20	1	Jean
2015	08	17	3	Kevin

CODE

```
CREATE TABLE NoSQL (
    ID          BIGINT,
    Année       INT,
    Mois        INT,
    Jour        INT,
    Nom         STRING
)
PARTITIONED BY (ID INT, Année
INT, Mois INT) ;
```

NoSQL – Orienté colonne



On voit bien le rôle de map reduce dans cette partie, qui « compte » le nombre de données similaires (par exemple 3*2015 et 2*07 dans ce tableau). Cela permet de de savoir la taille du bloc de données dans le tableau.

Avantages :

- Organisation en colonne pour moins de redondance de code, donc plus d'espace
- **Accès rapide par clé ou recherche de donnée**
- **Création d'indexes**, jointures et recherches faciles
- SQL-like : Facile à programmer et à prendre en main
- Pas besoin de s'occuper de MapReduce, Hive s'en charge

2. ZOOKEEPER : LE GARDIEN DE L'ECOSYSTEME

Zookeeper est un **service** (application à installer) d'Apache et est **essentiel** dans une architecture Hadoop.

Le rôle de Zookeeper est d'assurer la **cohérence de l'application**, de la rendre **simple d'utilisation** et de développement en mettant en relation ses éléments. Il agit au niveau de : la **configuration**, **message queue**, **les notifications** et la **synchronisation**.

Les données enregistrées dans le HDFS et les applications (Hive, Pig, MapReduce, HBase …) sont sur plusieurs machines. Leurs configuration doit donc être synchronisée afin d'avoir les mêmes versions et les mêmes configuration.

Zookeeper permet de **synchroniser ces configurations dans tout le cluster**. Cela évite de configurer les applications une par une sur chaque node.

LIENS

Tutoriel d'Hadoop : https://www.youtube.com/watch?v=2jzpkdYHwSQ

3. HBASE

HBase est une map triée permettant des accès direct rapide et d'écrire en masse à des données. Cette solution est OpenSource. On l'utilise par exemple pour des cas de messagerie (Facebook messages).

HBase est notamment utile pour l'accès direct et les colonnes vide d'une donnée ne demande pas de temps. La lecture de colonnes éparse.

4. AMBARI : LE GESTIONNAIRE DES NOEUDS

Appache Ambari est un service permettant de gérer le cluster. A l'aide de ce framework on peut ajouter, supprimer, configurer les nodes.

Apache ambari dispose d'une api rest, d'une interface graphique permettant de voir les informations du cluster.

Example - Get all hosts with less than 2 cpu or host status != HEALTHY

```
GET /api/v1/clusters/c1/hosts?Hosts/cpu_count<2|Hosts/host_status!=HEALTHY
```

Example - Create the HDFS service.

```
POST /clusters/c1/services/HDFS
```

Response

```
201 Created
```

LIENS

http://ambari.apache.org/

https://github.com/apache/ambari/blob/trunk/ambari-server/docs/api/v1/index.md

5. OOZIE / AZKABAN / LUIGI MESOS : L'ACCUEIL DES TACHES

Oozie est une application web java contenue dans un conteneur de servlet et qui permet **d'ordonnancer** et de **rediriger** les requêtes reçues.

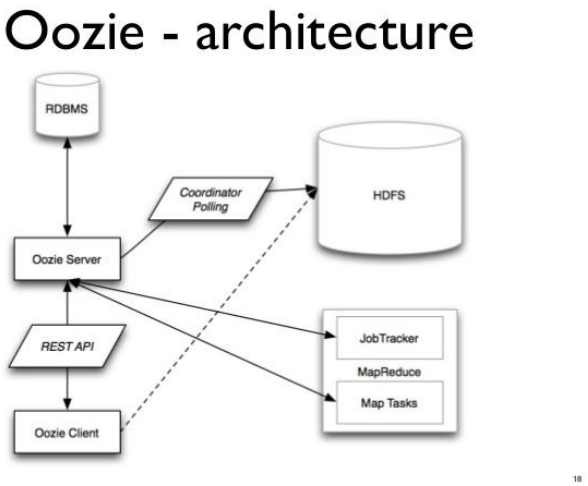
En Résumé, Oozie permet de **rediriger les tâches vers le bon outil** Hadoop.

Il permet aussi d'exécuter des tâches **en fonction de dates** (le 22/02 à 17h15) , de **fréquences** (toutes les 5 minutes) et **en fonction des requêtes** qu'il reçoit (« GET localhost/oozie.data?customerID=4 »).

Il permet aussi de définir le fait qu'un job doit être exécuter sur les données avant d'entrer dans un module. Si les données doivent être transformées avant d'être enregistrées dans le HDFS, on définit la propriété dans Oozie.

Il fait exécuter ses tâches par MapReduce, Pig, Hive, au HDFS, Hadoop Streaming ou peut exécuter votre code Java et Oozie Sub-Workflow (Nous ne reviendrons pas sur ces derniers termes mais en gros, il peut donner des tâches à faire à tous les composants de votre installation hadoop et les mettres dans une file pour les traiter tour à tour).

1 OOZIE ARCHITECTURE

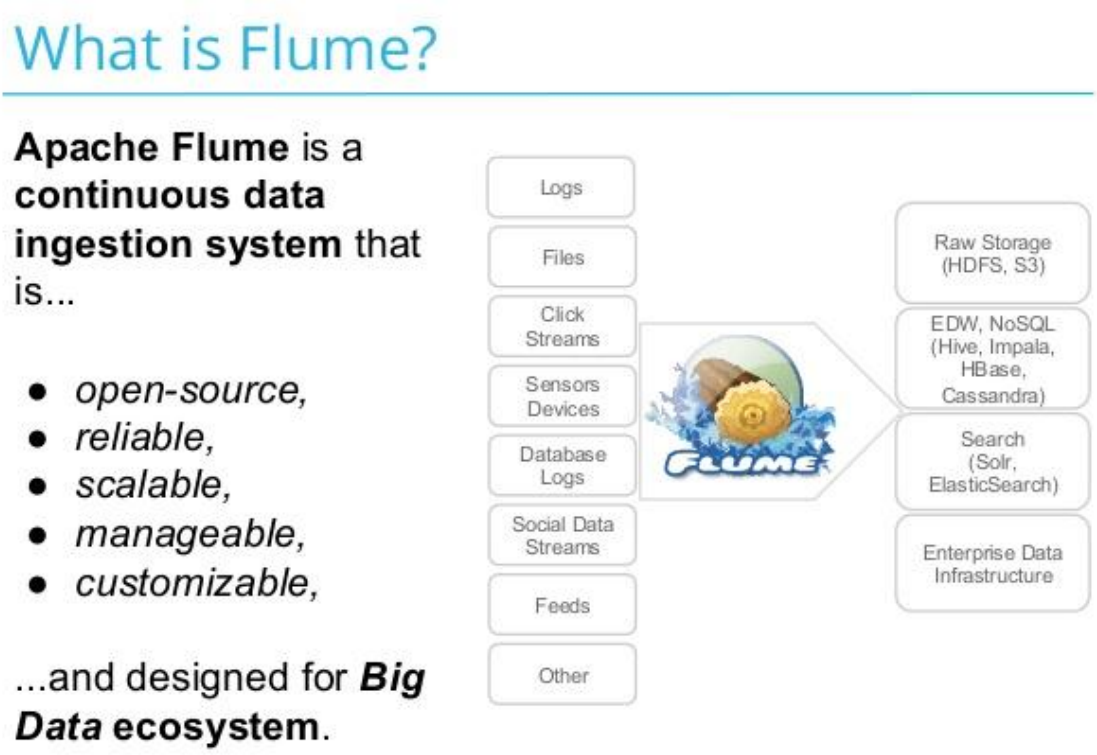


LIENS

Présentation Oracle : http://fr.slideshare.net/jcrobak/data-engineermeetup-201309

6. FLUME / SCRIBE / KAFKA / CHUKWA : RECEPTION DE FLUX CONSTANT DE DONNEES

Flume est un framework java qui permet de récupérer des **données en streaming** (flux direct), de différentes

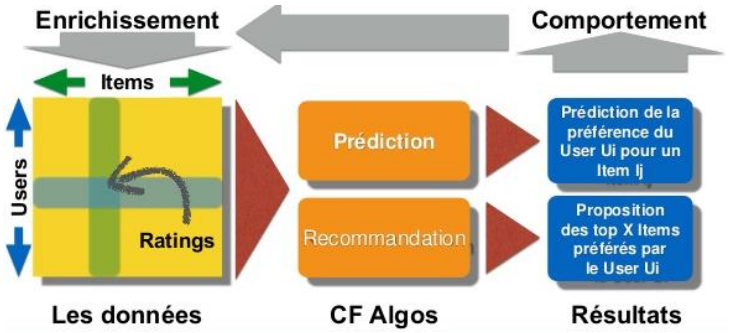


LIENS

Présentation par Yahoo! : http://fr.slideshare.net/ydn/flume-hug?qid=32c4242d-eb0e-4d4f-860b-65576d97b3c5&v=default&b=&from_search=2

7. MAHOUT : PREDICTION / ENRICHISSEMENT DES DONNEES

Bibliothèque java open source de Machine Learning. Ses calculs sont beaucoup plus performant que MapReduce.



X. LES DISTRIBUTIONS HADOOP

Il existe trois principales distribution pour hadoop : Cloudera, Hortonworks et MapR.

Ces distributions consistent à faciliter le développement, rajouter des outils, rendre plus sur l'application …
Elles englobent tous les composants qu'elles proposent. Ces solutions ont cependant un coût.

Key features of three popular Hadoop distributions

The values in the cells of the table refer to the versions of the corresponding components available in a particular Hadoop distribution. For performance comparisons, see our Hadoop Distributions: Cloudera vs. Hortonworks vs. MapR study.

Component (name and version)	Solution title	Hortonworks HDP	Cloudera CDH4.3	MapR M3 v3.0
FILE SYSTEM		HDFS 1.2.0	HDFS 2.0.0	MapR-FS
• non-Hadoop access		NFSv3	Fuse-DFS v2.0.0	Direct Access NFS
• Web access	REST HTTP API	WebHDFS	HttpFS	*
MAPREDUCE		1.2.0	0.20.2	**
• software abstraction layer	Cascading	x	x	2.1
NON-RELATIONAL DATABASE	Apache HBase	0.94.6.1	0.94.6	0.92.2
METADATA SERVICES	Apache HCatalog	***Hive	0.5.0	0.4.0
SCRIPTING PLATFORM	Apache Pig	0.11	0.11.0	0.10.0
• data analysis framework	DataFu	x	0.0.4	x
DATA ACCESS AND QUERY	Apache Hive	0.11.0	0.10.0	0.9.0
WORKFLOW SCHEDULER	Apache Oozie	3.3.2	3.3.2	3.2.0
CLUSTER COORDINATION	Apache Zookeeper	3.4.5	3.4.5	3.4(?)
BULK DATA TRANSFER BETWEEN RELATIONAL DATABASES AND HADOOP	Apache Sqoop	1.4.3	1.4.3	1.4.2
DISTRIBUTED LOG MANAGEMENT SERVICES	Apache Flume	1.3.1	1.3.0	1.2.0
MACHINE LEARNING AND DATA ANALYSIS	Apache Mahout	0.7.0	0.7	0.7
HADOOP UI	Hue	2.2.0	2.3.0	-
• data integration service	Talend Open Studio for Big Data	5.3	x	x
CLOUD SERVICES	Whirr	x	0.8.2	0.7.0
PARALLEL QUERY EXECUTION ENGINE		Tez (Stinger)	Impala	****
FULL-TEXT SEARCH	Search		0.1.5	
ADMINISTRATION		Apache Ambari	Cloudera Manager	MapR Control System
• installation		Apache Ambari	Cloudera Manager	-
• monitoring		Ganglia	x	x
		Nagios	x	x
• fine-grained authorization	Sentry		1.1	
NON-MAPREDUCE TASKS	YARN	2.0.4	2.0.0	-

The functionality of Hortonworks, Cloudera, and MapR (Source: Altoros)

NOTES:

* via NFS
** MapR has its own Hadoop compatible MapReduce implementation
*** HCatalog has been merged with Hive, the latest standalone release version was 0.5.0
**** Apache Drill is at an early development stage
x - available, but not mentioned in the distribution documentation / requires manual installation or additional configuration.

XI. LES CONCURENTS HADOOP

Il existe des solutions NoSql différentes d'Hadoop qui sont fiables et très efficaces. Notamment Cassandra (qui peut s'intégrer à Hadoop) et qui est similaire mais orienté Ecriture plus que lecture. C'est le plus gros concurrent de HBase.

XII. LIENS UTILES / REMARQUES

Classement des solutions de Base de données: http://db-engines.com/en/ranking

Configuration d'Hadoop

Introduction à Appache Hadoop

Ecosystème Hadoop (20 minutes)

Installation d'un multi-node (et single node)

Si vous avez des remarques : alex.tremoceiro@sfr.com

XIII. A FAIRE

~~Peux-tu préciser le rôle du « name node » et du « edge node » ?~~

~~Et les notions de répartitions de charge et de redondance (primary, secondary)~~

~~1 data node = 1 serveur + 1 disque, ou s'agit-il d'une notion plus abstraite ? (2 data node sur le même serveur / même disque, même si l'intérêt peut en être réduit).~~

~~Sous quel format sont déposés les données ?~~

~~Comment est fait l'indexation ?~~

~~Décrire plus en détail Hive, qui semble être intéressant pour SUMO.~~

~~Mapper les outils hadoop sur le fonctionnel SUMO.~~

~~Dans map reduce, le rôle de « map » et celui de « reduce ».~~

Sait-on estimer grosse maille l'impact sur les performances du dépôt de fichiers zippés plutôt que de fichiers en clair ?

➔ Possiblité de le faire, mais peu d'informations concernant les indexes. Dans Cassandra les indexes ne sont pas compressé mais je ne sais pas pour hadoop (il ne me semble pas).

XIV. QUESTIONS ORACLE/ERICSSON

(si entretien Cloudera) : Proposez-vous une solution Hardware également ?

Cassandra ne serait pas mieux adaptée ?

Le edge node est-il indispensable ?

Un checkpoint node ou back-up node serait-il nécessaire ?

Quels outils sont utilisés et proposés ?

Quels type de support est proposé ?

Quels sont les licence utilisées et sont-elles à ajouter au prix (je suppose que non) ?

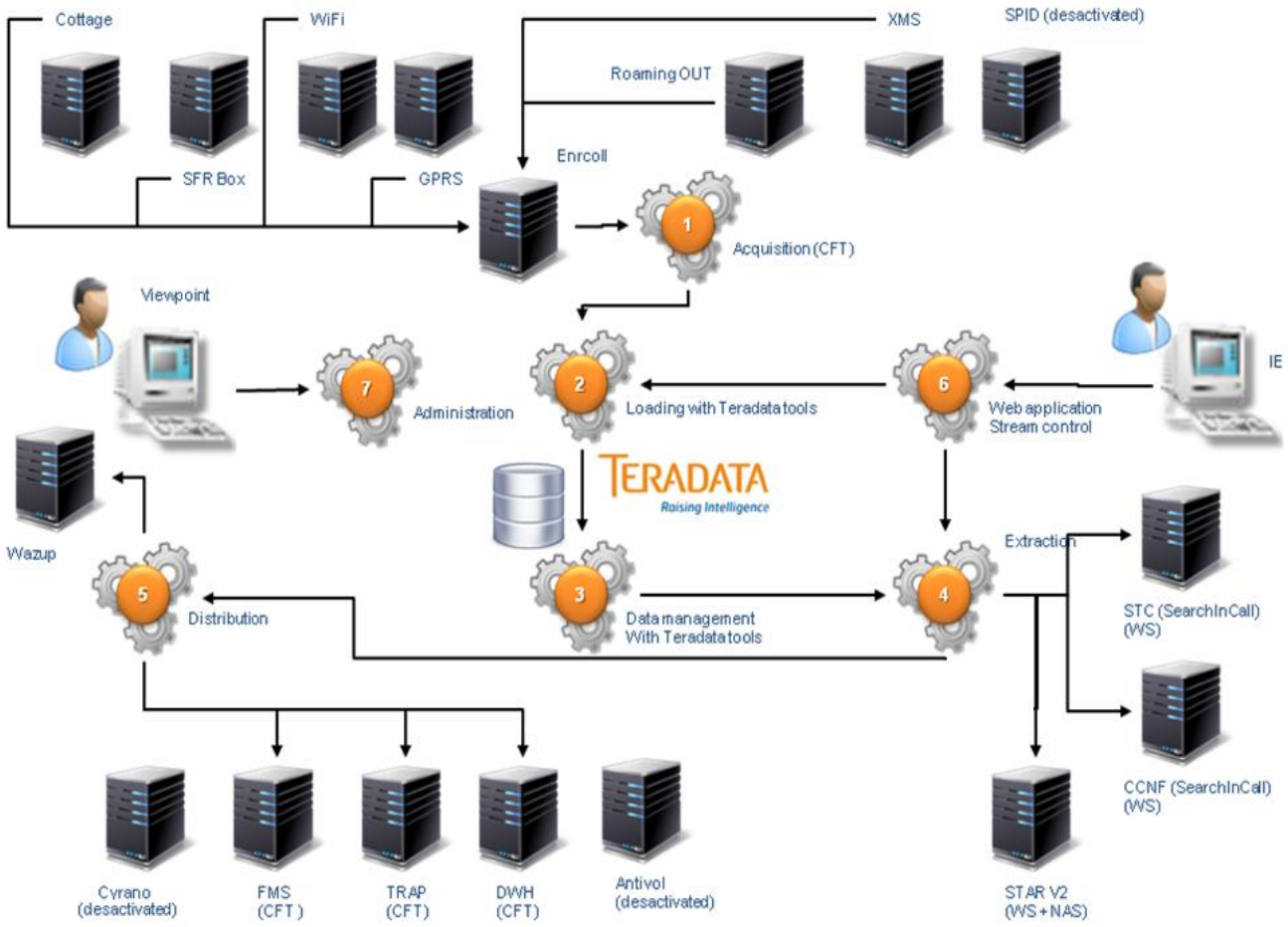
Comment se passe la formation avec les équipes qui travaillent sur le projet ?

Pourquoi ne pas prendre du hardware plus petit et moins cher en plus grosse quantité ?

Les outils NoSQL sont déjà très complets, quel serait donc l'avantage d'une surcouche oracle/cloudera ?

SUMO

XV. SUMO ACTUEL



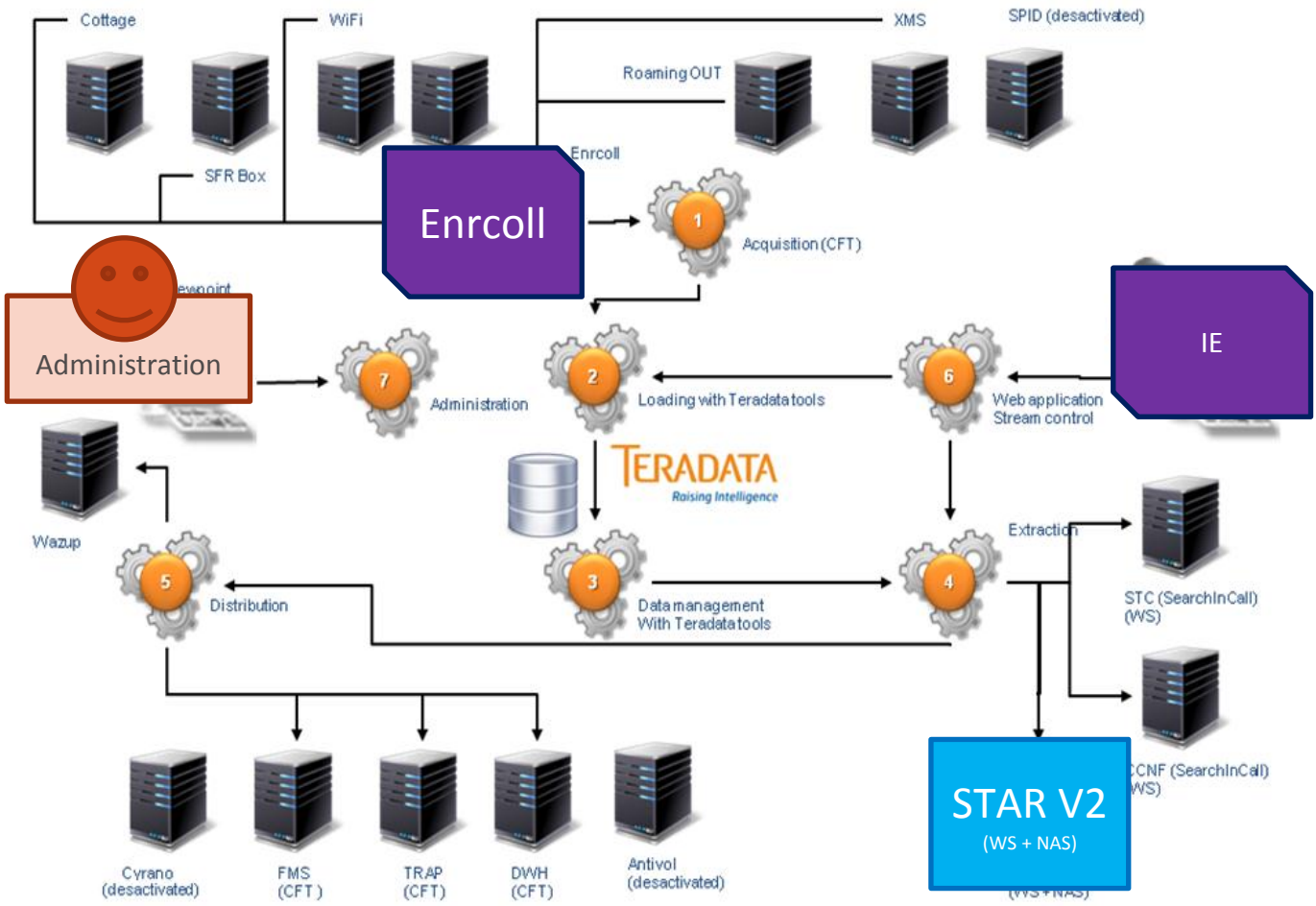
XVI. LEGENDE

Service SUMO

Approvisionnement en données vers SUMO

Requêtes venant de l'extérieur

(Schéma sur la prochaine page)



Je ne sais pas pour l'instant :

Comment marche la distribution (5 sur la dernière page) . Les données sont-elles mises en attente avant l'envoi ?

A quoi correspond IE exactement ? J'ai supposé qu'il s'agissait d'un flux constant de données.

XVII. SUMO HADOOP