Université d'Évry-Val d'Essonne

université PARIS-SACLAY

Master 1 Informatique & Miage

Bases de Données Approfondies – 2e partie

Théorie et pratique de NoSQL

Philippe Declercq

2022/2023

Plan du cours et des TD

1	Introduction au NoSQL
2	Stockage et manipulation de données avec NoSQL
3	Réplication des données
4	Partitionnement des données
5	Recherches et calculs en NoSQL
6	Et l'avenir ?
TD1	Installation et manipulations simples NoSQL
TD2	Réplication et partitionnement
TD3	Recherche d'information – Moteurs de recherche

Université d'Évry-Val d'Essonne

université PARIS-SACLAY

Théorie et pratique de NoSQL

Chapitre 5

Recherches et calculs en NoSQL

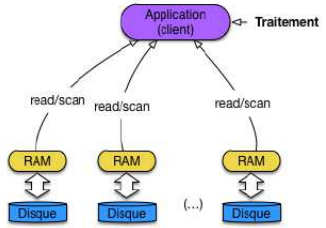
Philippe Declercq

2022/2023

Recherches et calculs en NoSQL

- Les bases NoSQL sont conçues pour accueillir un très grand volume de données, qu'elles gèrent en mettant en place des mécanismes de réplication et partitionnement. La recherche d'informations dans ces bases doit être performante : on veut bien souvent du temps réel, ou presque.

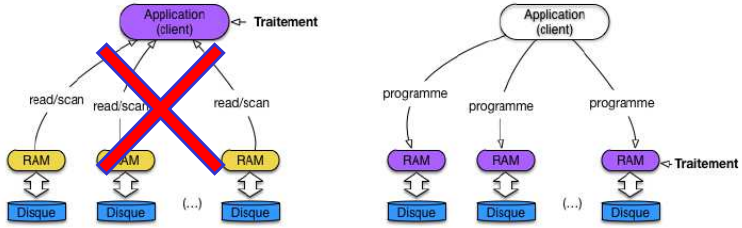
- Si on applique une « stratégie » classique de type client-serveur, l'application client va recevoir des données du serveur et y appliquer ses calculs :



Recherches et calculs en NoSQL

- Les bases NoSQL peuvent utiliser les capacités de calcul de chaque partition qui héberge des données.

- Avantages :
  - Les données sont traitées localement, on distribue un programme de calcul plutôt que de transférer de gros volumes de données,
  - On profite des serveurs pour effectuer des calculs en parallèle.



Recherches et calculs en NoSQL

- Pour écrire une application distribuée basée sur ce principe, il faut :
  1. implanter la logique de l'application, autrement dit le traitement particulier qui peut être plus ou moins complexe,
  2. concevoir la parallélisation de cette application, sous la forme d'une exécution concurrente coordonnant plusieurs machines et assurant un accès à un partitionnement de la collection traitée,
  3. gérer la reprise sur panne dans un environnement potentiellement instable.

- Des mécanismes et des solutions qui prennent en charge les points 2 et 3 ont été conçus.

- C'est en particulier l'objectif du framework d'exécution « MapReduce ».

Recherches et calculs en NoSQL - MapReduce

- « MapReduce » a été mis au point et breveté par Google en 2004.

- « System and method for efficient large-scale data processing »

- Ce modèle de programmation a été adopté par les sociétés concernées par le traitement de très grands volumes de données.

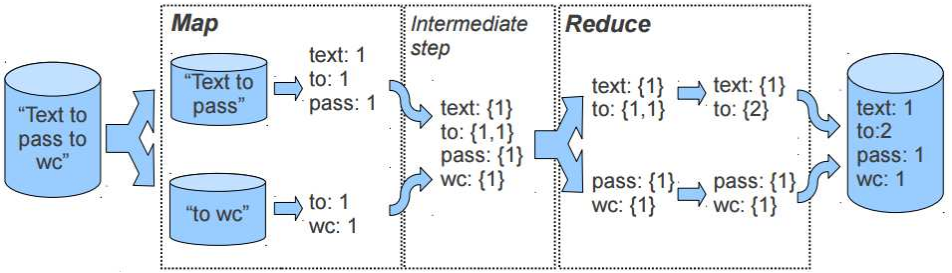
Recherches et calculs en NoSQL - MapReduce

- « MapReduce » permet :
  - de répartir une charge de traitements de données sur plusieurs nœuds,
  - de gérer, de façon transparente, l'infrastructure matérielle,
  - d'assurer la scalabilité : les traitements étant répartis, une augmentation du volume des données ne dégrade par les performances, lorsque ces données sont distribuées.

- Principe général de « MapReduce » : étant donné une collection d'items, on applique à chaque item un processus de transformation individuelle (Map) qui produit des valeurs intermédiaires étiquetées. Ces valeurs intermédiaires sont regroupées par étiquette et soumises à une fonction d'assemblage appliquée à chaque groupe (Reduce).

- Utiliser mapReduce nécessite de définir, selon la recherche souhaitée :
  - La fonction map : prend en entrée une paire (clé1, valeur1) et produit une liste de paires intermédiaires (clé2, valeur2).
  - La fonction reduce : prend en entrée une paire (clé2, liste(clé2, valeur2)) et produit valeur3.
- Le reste est pris en charge par le système qui implémente mapReduce.

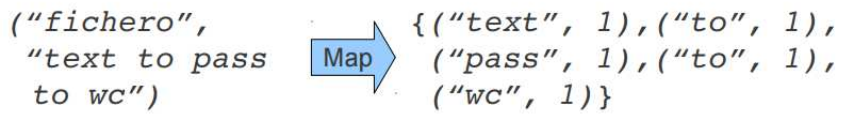
- Exemple : calcul des occurrences de mots dans une phrase :



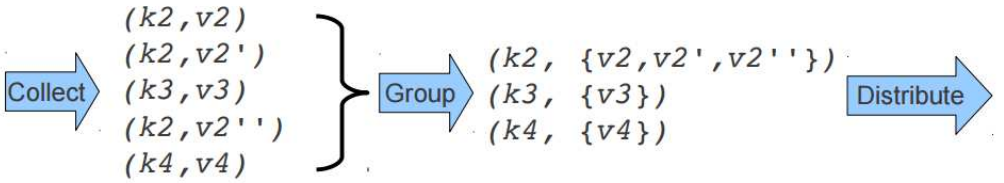
- La fonction map peut être programmée comme cela :

```
map(String key, String value) {
// key: doc name, value: doc text
for each word w in value:
  EmitIntermediate(w, "1");
}
```

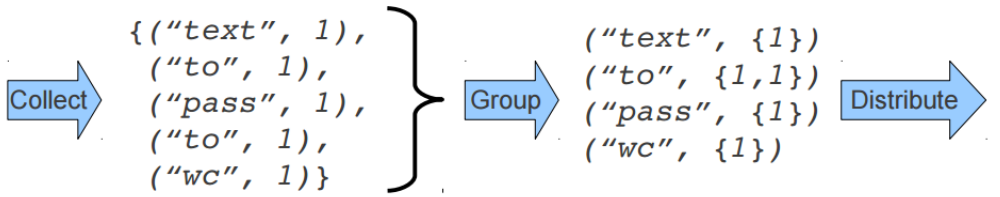
- Ce qui donne :



- L'étape intermédiaire consiste à :
  - Collecter les résultats des calculs de map,
  - Grouper les paires par clé,
  - Distribuer les clés aux nœuds qui exécutent Reduce.



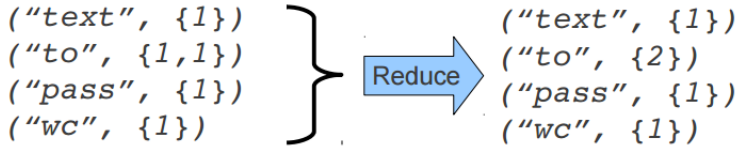
- Ce qui donne :



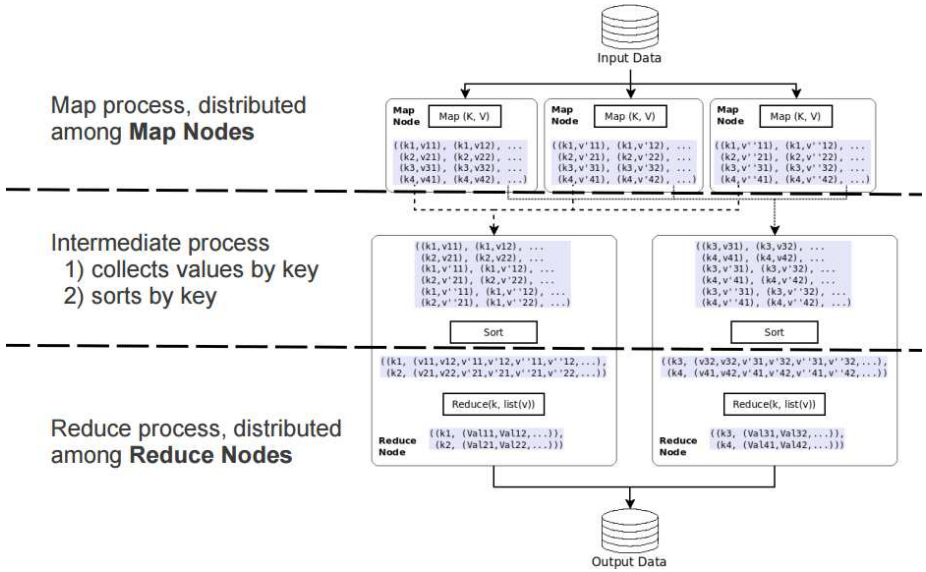
- La fonction reduce peut être programmée comme cela :

```
reduce(String key, List values) {
// key: word, value: list of counts
int result = 0;
for each v in values:
  result +=ParseInt(v);
Emit(key, result);
}
```

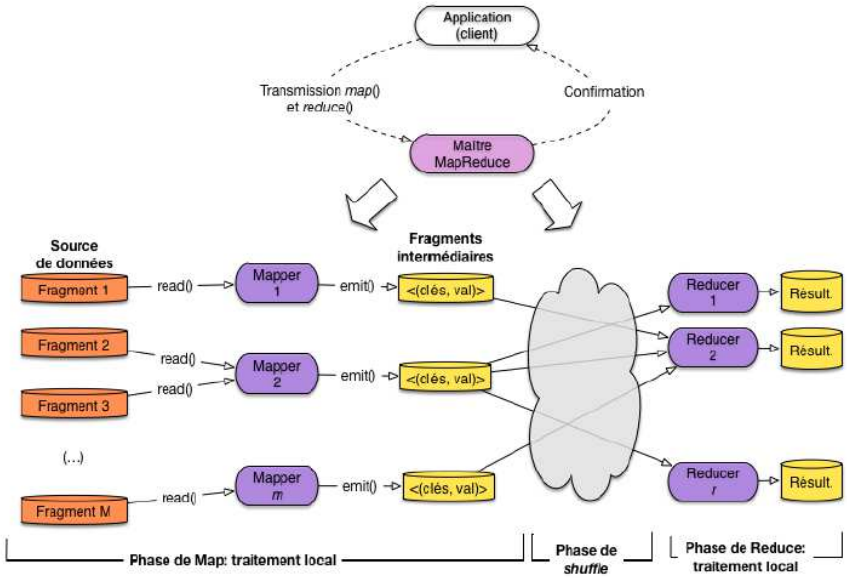
- Ce qui donne :



- En synthèse :



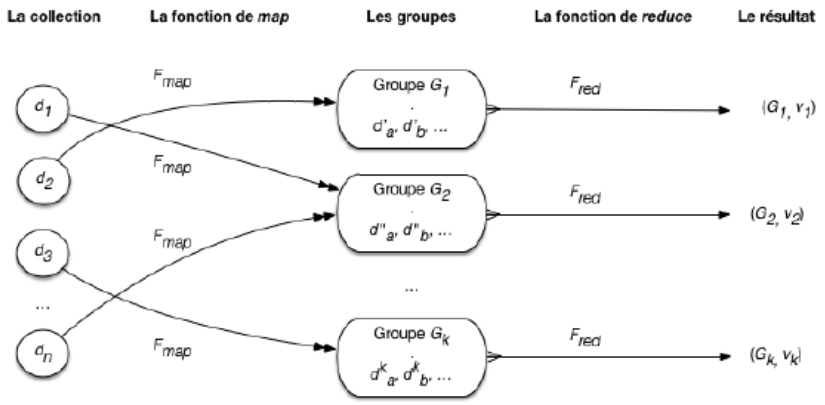
- En environnement distribué :



- mapReduce ne permet pas toutes les recherches ! Il n'offre pas la richesses du langage d'interrogation SQL.
- Les opérations map et reduce ne permettent en effet que des manipulations relativement simples.

- Utilisations courantes :
  - Recherche d'éléments qui correspondent à certains critères
    - Map → Recherche élément par élément, émettre si l'élément correspond
    - Reduce → Identité
  - Dénombrement de la fréquence d'accès à des URL: compter dans des logs les nombres d'accès par URL
    - Map → A chaque URL , émettre <URL, 1>
    - Reduce → Regoupe ensemble les valeurs, émettre <URL, sum>

- Sous mongodb, les items vont correspondre à des documents distribués sur différents nœuds.



- Soit la collection « cours » sous mongodb :

cours 0 sec.

```
/* 1 */
{
    "_id" : ObjectId("56509aa6769a89f17ce001bf"),
    "nom" : "Reseaux avancés",
    "heures" : 18,
    "enseignant" : "Toto"
}

/* 2 */
{
    "_id" : ObjectId("56509abe769a89f17ce001c0"),
    "nom" : "Gestion de projets",
    "heures" : 15,
    "enseignant" : "Titi"
}

/* 3 */
{
    "_id" : ObjectId("56509ccc769a89f17ce001c2"),
    "nom" : "Programmation",
    "heures" : 12,
    "enseignant" : "Toto"
}
```

- Etablir un document par enseignant, avec la liste des enseignements qu'il dispense.

- Définition des fonctions map et reduce :

```
var mapEns = function () {emit(this.enseignant, this.nom);};
var reduceEns = function(enseignant, cours) {
    var res = new Object();
    res.enseignant = enseignant;
    res.cours = cours;
    return res;
};
```

- Lancement du calcul :

```
db.cours.mapReduce(mapEns,reduceEns,{out : {"inline" : 1}})
```

- Résultat :

0.119 sec.

```
/* 1 */
{
    "results" : [
        {
            "_id" : "Titi",
            "value" : "Gestion de projets"
        },
        {
            "_id" : "Toto",
            "value" : {
                "enseignant" : "Toto",
                "cours" : [
                    "Reseaux avancés",
                    "Programmation"
                ]
            }
        }
```


- MapReduce peut être utilisé en traitement asynchrone pour établir des statistiques utilisateurs à partir d'informations de sessions.

- Soit la collection « sessions » créée sous mongodb :

```
db.sessions.insert( { userid: "a", ts: ISODate('2016-12-03 14:17:00'), length: 95 } );
db.sessions.insert( { userid: "b", ts: ISODate('2016-12-03 14:23:00'), length: 110 } );
db.sessions.insert( { userid: "c", ts: ISODate('2016-12-03 15:02:00'), length: 120 } );
db.sessions.insert( { userid: "d", ts: ISODate('2016-12-03 16:45:00'), length: 45 } );
db.sessions.insert( { userid: "a", ts: ISODate('2016-12-04 11:05:00'), length: 105 } );
db.sessions.insert( { userid: "b", ts: ISODate('2016-12-04 13:14:00'), length: 120 } );
db.sessions.insert( { userid: "c", ts: ISODate('2016-12-04 17:00:00'), length: 130 } );
db.sessions.insert( { userid: "d", ts: ISODate('2016-12-04 15:37:00'), length: 65 } );
db.sessions.insert( { userid: "d", ts: ISODate('2016-12-04 19:45:00'), length: 22 } );
```

- On souhaite avoir, par utilisateur, le temps total des sessions et le nombre de connexions.

- Définition des fonctions de Map et Reduce :
  - Map : créer des paires intermédiaires, avec pour clé « userid », et pour valeurs le userid, la durée de connexion et la valeur 1 correspondant au décompte de la connexion :

```
var mapFunction = function() {
        var key = this.userid;
        var value = {
                userid: this.userid,
                tempstotal: this.length,
                nombre: 1 };
        emit( key, value );
    };
```

  - Reduce : sommer sur la clé les durées de connexion et les décomptes de connexion :

```
var reduceFunction = function(key, values) {
        var reducedObject = {userid: key, tempstotal: 0, nombre:0 };
        values.forEach( function(value) {
                reducedObject.tempstotal +=
    value.tempstotal;
                reducedObject.nombre += value.nombre;  } );
        return reducedObject;
    };
```


- Lacement du traitement :

```
db.sessions.mapReduce(mapFunction, reduceFunction, {out: "session_stat"})
```

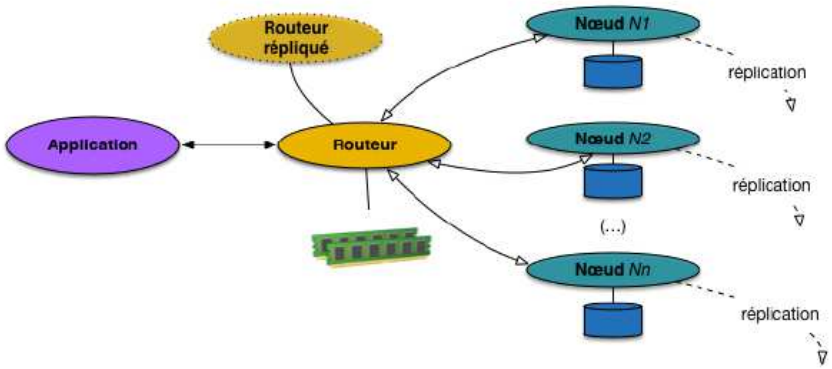

```
/* 1 */
{
    "_id" : "a",
    "value" : {
        "userid" : "a",
        "tempstotal" : 200.0000000000000000,
        "nombre" : 2.0000000000000000
    }
}

/* 2 */
{
    "_id" : "b",
    "value" : {
        "userid" : "b",
        "tempstotal" : 230.0000000000000000,
        "nombre" : 2.0000000000000000
    }
}
```


- On peut, périodiquement, lancer le traitement sur les nouvelles informations de sessions, pour compléter les résultats :

```
db.sessions.mapReduce( mapFunction, reduceFunction, {
        query: { ts: { $gt: ISODate('2016-12-04 00:00:00') } },
        out: { reduce: "session_stat" } });
```


- Sous mongodb, en configuration distribuée, le lancement de mapReduce se fait sur le routeur.

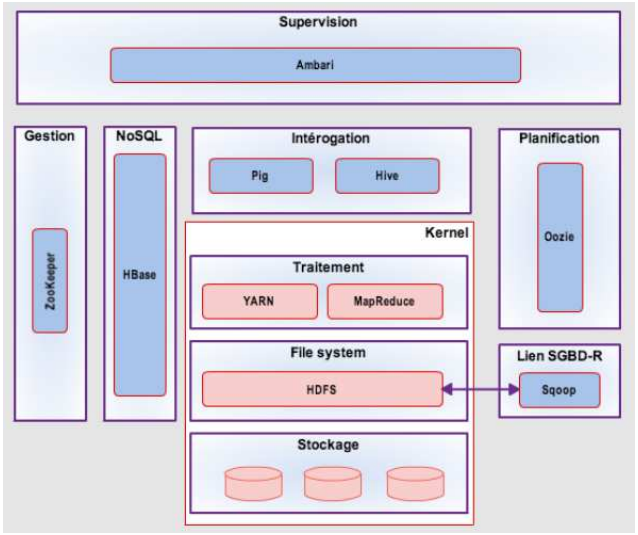


- Le résultat doit apparaître dans une collection … partitionnée sur les différents nœuds !

- Hadoop (« High-availability distributed object-oriented platform ») est un ensemble de projets open source coordonnés par l'Apache Software Foundation, destinés à faciliter la création d'applications qui opèrent sur un grand volume de données, distribuées.

- Hadoop propose, entre autres :
  - un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File System),
  - un système d'analyse des données appelé MapReduce.

- Vue d'ensemble de la plate-forme Hadoop :

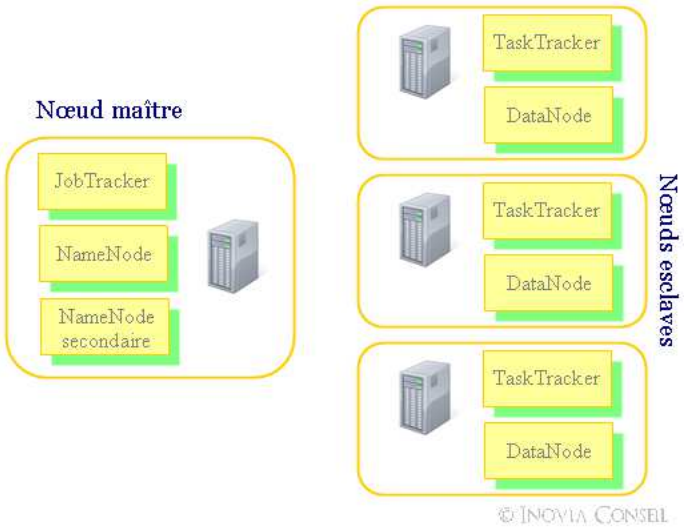


http://blog.ippon.fr

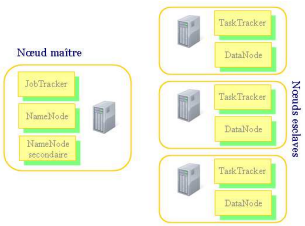
- La vocation première de Hadoop est d'implémenter des traitements batchs performants qui impliquent un volume de données très important.

- Usages :
  - Détection de fraudes,
  - Gestion du risque,
  - Moteurs de recommandations,
  - Analyse de données,
  - Décisionnel…

- Hadoop fonctionne en environnement distribué maître-esclave :



- La gestion des données est basée sur le système de fichier distribué HDFS. Le NameNode gère l'arborescence du système de fichiers et la localisation des blocs de données répartis. Les DataNode stockent et restituent les blocs de données.
- Le contrôle des processus de traitement : le JobTracker coordonne l'exécution des jobs sur l'ensemble des noeuds. Il communique avec les TaskTrackers en leur attribuant des tâches d'exécution (map ou reduce). Il permet d'avoir une vision globale sur la progression ou l'état du traitement distribué. Les TaskTrackers exécutent les tâches.



- L'API java Hadoop fournit des interfaces qu'il faut ensuite implémenter.

- Exemple :
  - map :

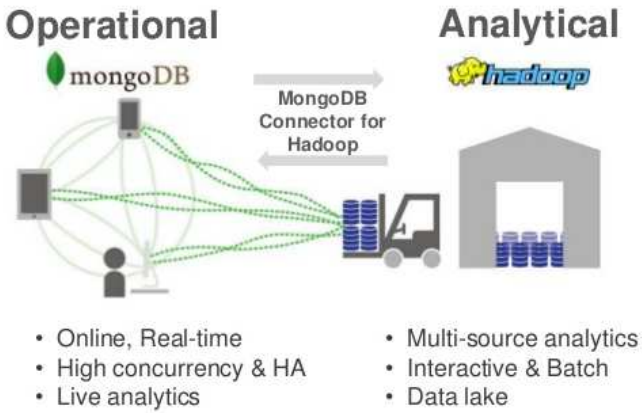
```
.class WordCountMapper extends MapReduceBase implements Mapper {
.  private final static IntWritable mDefOcc = new IntWritable(1);
.  private Text mWord = new Text();
.  public void map(LongWritable key, Text value, OutputCollector output, Reporter
reporter) throws IOException {
.    String lLine = value.toString();
.    StringTokenizer lIt = new StringTokenizer(lLine);
.    while (lIt.hasMoreTokens()) {
.      mWord.set(lIt.nextToken());
.      output.collect(mWord, mDefOcc);
.    }
```

  - reduce :

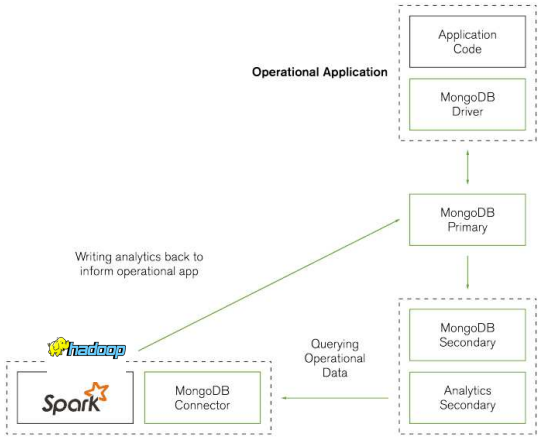
```
.public class WordCountReducer extends MapReduceBase implements Reducer {
.  public void reduce(Text key, Iterator values, OutputCollector output, Reporter
reporter) throws IOException {
.    int nbOcc = 0;
.    while (values.hasNext()) {
.      nbOcc += values.next().get();
.    }
.    output.collect(key, new IntWritable(nbOcc));
.  }
.}
```


- Hadoop peut s'utiliser avec d'autres systèmes de stockage des données que HDFS.

- Avec mongodb par exemple :

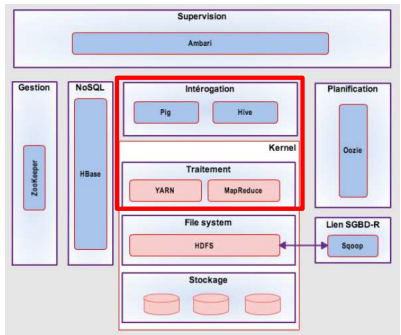


- Hadoop pour ses concurrents utilisent en général une réplication des données pour exécuter ses traitements.
- Le résultat des traitements peut être directement mis à disposition des applications opérationnelles.



- La complexité de programmation des requêtes MapReduce a conduit au développement de langages de plus haut niveau basés sur MapReduce.

- Exemples : DryadLINQ, Sawzall, Pig, Hive.



Recherches et calculs en NoSQL

- Ces solutions fournissent des solutions de requêtages très puissantes sur de grandes quantités de données, distribuées.
- Ces solutions s'apparentent alors aux systèmes décisionnels.

Recherches et calculs en NoSQL - Pig

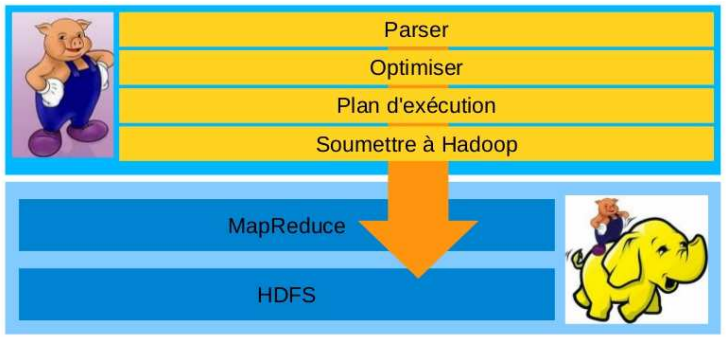
- Pig est une plate-forme pour la création de programmes utilisables avec Hadoop.
- Développée par Yahoo en 2006 pour implémenter des traitements MapReduce sur des volumes de données très importants. Repris par la fondation Apache depuis.
- Pig Latin est le langage de programmation de Pig. Il permet de s'affranchir de la complexité de MapReduce pour se rapprocher de la syntaxe de SQL.

Recherches et calculs en NoSQL - Pig

- Exemple de code Pig Latin :

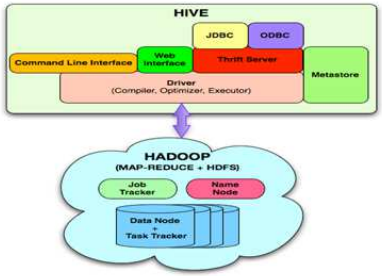
```
Lines = LOAD '/data/texts/*.txt' AS (line:chararray);
Words = FOREACH Lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
Groups = GROUP Words BY word;
Counts = FOREACH Groups GENERATE group, COUNT(Words);
STORE Counts INTO 'counts';
```

- Le code permet d'adresser des requêtes sur des données distribuées dans HDFS.



Recherches et calculs en NoSQL - Hive

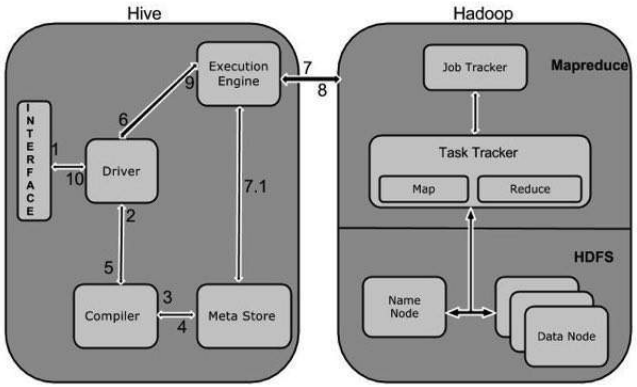
- Hive est une infrastructure construite sur Hadoop pour l'analyse de données.



- Initialement développé par Facebook, à présent repris par la fondation Apache.
- Propose un langage d'interrogation, HiveQL, qui propose la syntaxe d'interrogation du SQL.

Recherches et calculs en NoSQL - Hive

- Hive tranforme les requêtes HQL en MapReduce, et les transmet à un cluster Hadoop pour qu'elles soient exécutées.

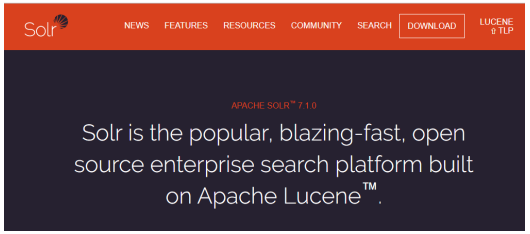


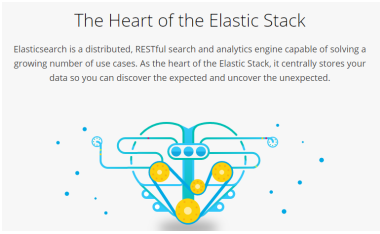
- Une base de données, Meta Store, permet de définir le méta-modèle utilisé par HQL, et sa transcription dans Hadoop.

Recherches et calculs en NoSQL – Moteurs de recherche

- Certaines solutions NoSQL se sont spécialisées et sont optimisées pour la recherche d'informations, et en particulier dans la recherche textuelle.

- Solr et elasticsearch sont les deux moteurs de recherche les plus répandus dans cette catégorie de solutions. Ils se basent tous les deux sur lucene distribué par la fondation Apache.





Recherches et calculs en NoSQL – Moteurs de recherche

- Cette technologie utilise le concept d'index inversé pour optimiser les recherches.

- Exemple :
  - Pour les 4 documents qui contiennent les phrases suivantes :
    - 1 : « Ce logiciel est très performant »
    - 2 : « La rapidité du logiciel est évidente »
    - 3 : « Cette phrase est évidente »
    - 4 : « Le logiciel est il gratuit ? »
  - Sera stocké :
    - logiciel : 1;2;4
    - performant : 1
    - évidente : 2;3
    - …

Recherches et calculs en NoSQL – Moteurs de recherche

- Exemple : recherche textuelle dans elasticsearch

- Insertion des textes dans elasticsearch :

- Elasticsearch permet des recherches :
  - Quelque soit les champs indexés,
  - Exactes sur un/des mots clés,
  - Approchées,
  - En affectant des poids différents aux critères de recherche,
  - En précisant le nombre maximal de résultats souhaités,
  - En précisant le temps maximal d'exécution de la recherche,
  - En retournant les résultats affectés d'un score,
  - ….

- En ce sens, on parle de recherche d'informations plus que d'interrogation d'une base de données !

- Les moteurs de recherche s'utilisent souvent en compléments de solutions de stockage de données.

- Exemples d'architecture :