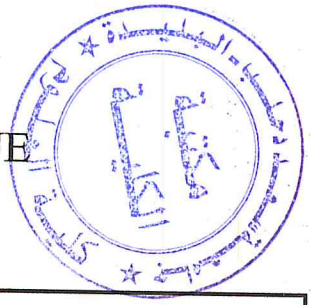
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB - BLIDA

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE



Mise en œuvre d'une plateforme d'intégration (ETL) pour le big data

Réalisé par : MOKEDDEM Oussama
MAMI Mohamed Nadjib

Encadré par : M. BALA Mahfoud

Soutenu le SAMEDI 29/06/2013 devant le jury constitué de :

Mme. OUAHRANI	USDB	Présidente
Mme. AZZOUZ M	USDB	Examinatrice
Mme. ARKAM	USDB	Examinatrice
M. BALA Mahfoud, Maître Assistant	USBD	Encadreur

Promotion : 2012 / 2013
Session : Juillet 2013

Sommaire

Liste des figures :

Résumé

ETL, acronyme de Extracting-Transforming-Loading (ou parfois datapumping), est un système de chargement de données depuis les différentes sources d'information de l'entreprise (hétérogènes) vers l'entrepôt de données (Base de données multidimensionnelle). Au fait, ce système ne se contente pas de charger les données, il doit les préparer et les normaliser (les filtrer, les nettoyer, les mettre dans un format approprié, les contextualiser, les homogénéiser, les agréger) et enfin les charger dans leur destination finale qu'est l'entrepôt de données. Un des défis majeurs pour le domaine de l'ETL est le traitement de très grandes quantités de données connues aujourd'hui sous le nom de Big Data. Déjà que ce processus est complexe vu la diversité et l'hétérogénéité des sources ainsi que la complexité des tâches ETL, il devra faire face à des données massives caractérisées par une volumétrie importante (PetaBytes, HexaBytes, ZettaBytes, …), de nouvelles structures et une exigence en termes de temps de traitement. Nous nous proposons de mettre en œuvre une plateforme d'intégration de données (ETL) pour l'entreposage et l'analyse en ligne destinée pour le Big Data dans un environnement basé sur le paradigme MapReduce destiné pour le traitement parallèle à grande échelle de données intensives sur un cluster d'ordinateurs. Le projet Hadoop de la fondation Apache étant une référence pour les frameworks MapReduce open source.

Mots clés : Entreposage de données, ETL, Traitement parallèle, Performance, MapReduce, Hadoop, Big Data.

Introduction Générale

1. Contexte général

Le phénomène du Big Data bouleverse les systèmes d'information. Les données se métamorphosent, prennent de nouvelles structures mais exigent aussi une rapidité en termes de traitements.

La capture des données s'effectue aujourd'hui dans tous les domaines à des quantités et à une vitesse démesurée où l'homme n'est pas le seul acteur. Des caméras, capteurs, GPS, scanneurs, … génèrent à chaque instant des quantités gigantesques d'informations. La popularité d'internet, du web et des Smartphones ont contribué aussi à l'avènement du Big Data.

Il est vrai que trop d'informations tue l'information mais l'exploitation de cet océan de données par des applications d'analyse pour la prédiction et l'aide à la décision produit des informations synthétisées et pertinentes.

C'est dans ce contexte que nous intervenons pour mettre en œuvre une plateforme d'intégration de données de type ETL pour l'entreposage et l'analyse en ligne destinée pour le Big Data. En effet, les données constituent la matière première utilisée par les systèmes décisionnels pour produire des informations pertinentes pour l'analyse et pour l'aide à la décision. Les processus ETL (Extracting-Transforming-Loading) sont chargés de capturer ces données n'importe où quelque soit leurs formats, leurs structures et leur volumétrie ; l'essentiel est que ces données ramènent de la valeur et de la pertinence au processus d'analyse. Cependant, les données de dimension Big Data mettent ces processus ETL en difficulté vu leurs tailles (mesurées en PetaBytes voire en HexaBytes et ZettaBytes) et leurs nouvelles structures surtout que celles-ci exigent des performances très importantes.

De nouveaux environnements et paradigmes ont vu le jour ; tels que l'informatique dans les nuages (cloud computing) et MapReduce. Ces environnements sont destinés essentiellement pour le traitement à grande échelle des données massives sur des infrastructures de type cluster.

Le but du travail proposé dans ce PFE est de mettre en œuvre une plateforme ETL intégrée basée entièrement sur le paradigme MapReduce, i.e. pour les trois phases du processus à savoir la phase E (Extracting), T (Transforming) et L (Loading). Nous voyons que MapReduce peut être une bonne base pour la parallélisation de processus ETL. Dans ETL, le traitement de données présente la propriété composable de telle sorte que le traitement des

dimensions et des faits peut être divisée en plus petites unités de calcul et les résultats partiels de ces unités de calcul peuvent être fusionnés pour constituer les résultats définitifs dans un DW(entrepôt de données). Ceci est conforme bien avec le paradigme MapReduce en terme de Map et Reduce.

2. Problématique

Le processus ETL est complexe par nature, ce qui explique la multiplication des phases ETL spécifiques à savoir la transformation, le nettoyage, le filtrage, l'agrégation et le chargement. Jusqu'à présent, les efforts d'amélioration des outils ETL et d'intégration de données se sont concentrés sur l'ajout de fonctionnalités, qui dépens de deux facteurs essentiels : la facilité d'utilisation et l'évolutivité sans perte de performance. Confrontés à la diversité, à la vélocité et aux volumes croissants de données – le fameux phénomène du « Big Data ». La complexité des tâches ETL, nouvelles structures et volumétrie des données sources ainsi que l'exigence en termes de temps d'exécution provoquent une nouvelle problématique dans le monde décisionnel : Comment améliorer les performances de l'ETL ?

3. Objectif

L'objectif prévu dans ce projet est de réaliser une plateforme d'intégration (ETL) pour les Big Data fonctionnant dans un environnement distribué parallèle basée sur MapReduce. L'environnement Hadoop retenu pour la mise en œuvre n'est pas adapté pour l'entreposage et l'analyse en ligne (OLAP), néanmoins il est très ouvert et générique. Pour implémenter un système ETL avec toutes les spécificités en termes de concepts décisionnels (cube de données, table de faits, table de dimension, PK, FK, ….), Nous devrons enrichir cet environnement par des fonctions (UDF) qui permettent de constituer un processus ETL de bout en bout. En ce qui concerne le traitement parallèle, nous devrons développer des algorithmes de partitionnement autres que l'algorithme classique supporté par Hadoop (basé sur la taille des partitions) et personnaliser les fonctions Map et Reduce de sorte à prendre en charge les transformations et agrégations prévues dans le processus ETL des utilisateurs.

4. Organisation du mémoire

Notre mémoire se divise en trois parties :

- **1ère partie :** Nous présentons les concepts des systèmes décisionnels ainsi que leurs architectures. Cette partie est constituée de deux chapitres :
  - 1ère chapitre : Nous présentons l'architecture de la chaine décisionnelle et les concepts fondamentaux du décisionnel.
  - 2ème chapitre : Ce chapitre présente le processus ETL de manière détaillée ainsi que l'impact de la volumétrie des données sur la performance de ce processus.
- **2ème partie :** Nous exposons le phénomène de Big Data ainsi le paradigme MapReduce avec le processus ETL. Cette partie est composée de trois chapitres :
  - 3ème chapitre : Qui présente le phénomène des Big Data et leurs origines, caractéristiques.
  - 4ème chapitre : Nous expliquons en détail le paradigme MapReduce.
  - 5ème chapitre : Nous allons présenter un travail existant d'un processus ETL dans un environnement parallèle MapReduce "ETLMR" et nous allons voir comment on peut appliquer la logique de MapReduce pendant les phases du processus ETL
- **3ème partie :** Nous présentons la conception et la mise en œuvre du système, cette partie comprend deux chapitres :
  - 6ème chapitre : Nous allons présenter l'architecture de notre système et une étude détaillée des différents modules du système.
  - 7èmechapitre : Nous allons exposer la manière de l'implémentation des différents modules figurant dans la conception du système.

Nous clôturons ce manuscrit par une conclusion et des perspectives pour des travaux futurs.

Partie 1 : Etat de l'art

Chapitre I : *Systèmes d'information décisionnels*

Chapitre II : *Processus ETL*

Chapitre I

Systèmes d'information décisionnels

Chapitre I : Systèmes d'information décisionnels

Le contexte dans lequel nous vivons est de plus en plus complexe. Les technologies de l'information nous génèrent une multitude de données comme jamais auparavant. Le problème n'est donc plus tant d'acquérir une masse de données, mais de l'exploiter. Pour cela il faut collecter de l'information de qualité, la normaliser, la classer, l'agréger, et l'analyser, pour l'exploiter afin d'en extraire de la valeur et de l'intelligence pour des fins d'analyse et d'aide à la décision en temps opportun. Dans ce but, il est nécessaire de mettre en place un système d'information particulier, appelé système décisionnel. Ce système doit permettre de présenter de manière simple les chiffres recueillis pour mettre en lumière la conjoncture actuelle et indiquer implicitement la voie à suivre.

Dans ce chapitre nous allons présenter les concepts de base des systèmes décisionnels ainsi que leurs architectures.

1. Systèmes d'information décisionnels

1.1 Définition

Le système décisionnelle (en anglais : BI pour *Business Intelligence*) est un système informatisé interactif aidant le décideur à manipuler des données et des modèles pour résoudre des problèmes mal structurés [01].

« L'informatique décisionnelle englobe les solutions informatiques apportant une aide à la décision avec, en bout de chaîne, rapports et tableaux de bord de suivi à la fois analytiques et prospectifs. Le but est de consolider les informations disponibles au sein des bases de données de l'entreprise.» [02].

1.2 L'aide à la décision

L'aide à la décision a pour objectif d'accompagner un ou plusieurs décideurs dans le processus de prise de décision. Elle permet aux acteurs concernés de spécifier leurs besoins par des processus de collecte, d'analyse et d'échange d'informations [05]

1.3 Systèmes OLAP

Dans la littérature, plusieurs définitions sont proposées pour les systèmes OLAP [08] [12]. Dans ces définitions, les caractéristiques de base sont la structure dimensionnelle des données, les données forment des points dans un espace à plusieurs dimensions, et l'interactivité de

l'interrogation afin de s'approcher de la perception du décideur et de l'aider au mieux dans son processus de prise de décision. Le système OLAP peut être défini comme suit :

Un système OLAP est un système d'information décisionnel qui organise les données dans un espace dimensionnel. Il regroupe un ensemble d'outils en interaction qui réalisent la synthèse dynamique, l'analyse interactive et l'agrégation d'un grand volume de données afin d'améliorer le processus de prise de décision.

Ce système réunit un ensemble de nouvelles fonctionnalités décrites par 12 règles [08]. Les principales caractéristiques extraites de ces règles sont :

- la vision dimensionnelle des données, la transparence entre l'outil de visualisation et l'espace de stockage des données dimensionnelles,
- l'interopérabilité (l'outil rend invisible à l'utilisateur l'hétérogénéité des données),
- la manipulation intuitive des données et la flexibilité des restitutions, le décideur dispos d'une interface ergonomique de consultation.

Les systèmes OLAP visent à combler les lacunes des systèmes transactionnels. En effet, une des principales caractéristiques des systèmes transactionnels, est une activité de modification et d'interrogation fréquentes et répétitives [05]. L'accès au système est réalisé par de très courtes transactions. Enfin, la plupart de ces systèmes ne conservent pas les évolutions des données manipulées, seules les versions courantes sont conservées.

1.4 La chaine décisionnelle

Une chaine décisionnelle est un ensemble de concepts, outils, méthodes, et technologies (logicielles et matérielles) qui, une fois mises en relation, permettent de créer de la connaissance et répondre aux besoins stratégiques de l'entreprise (dans le meilleur des cas). (Voir figure 1).

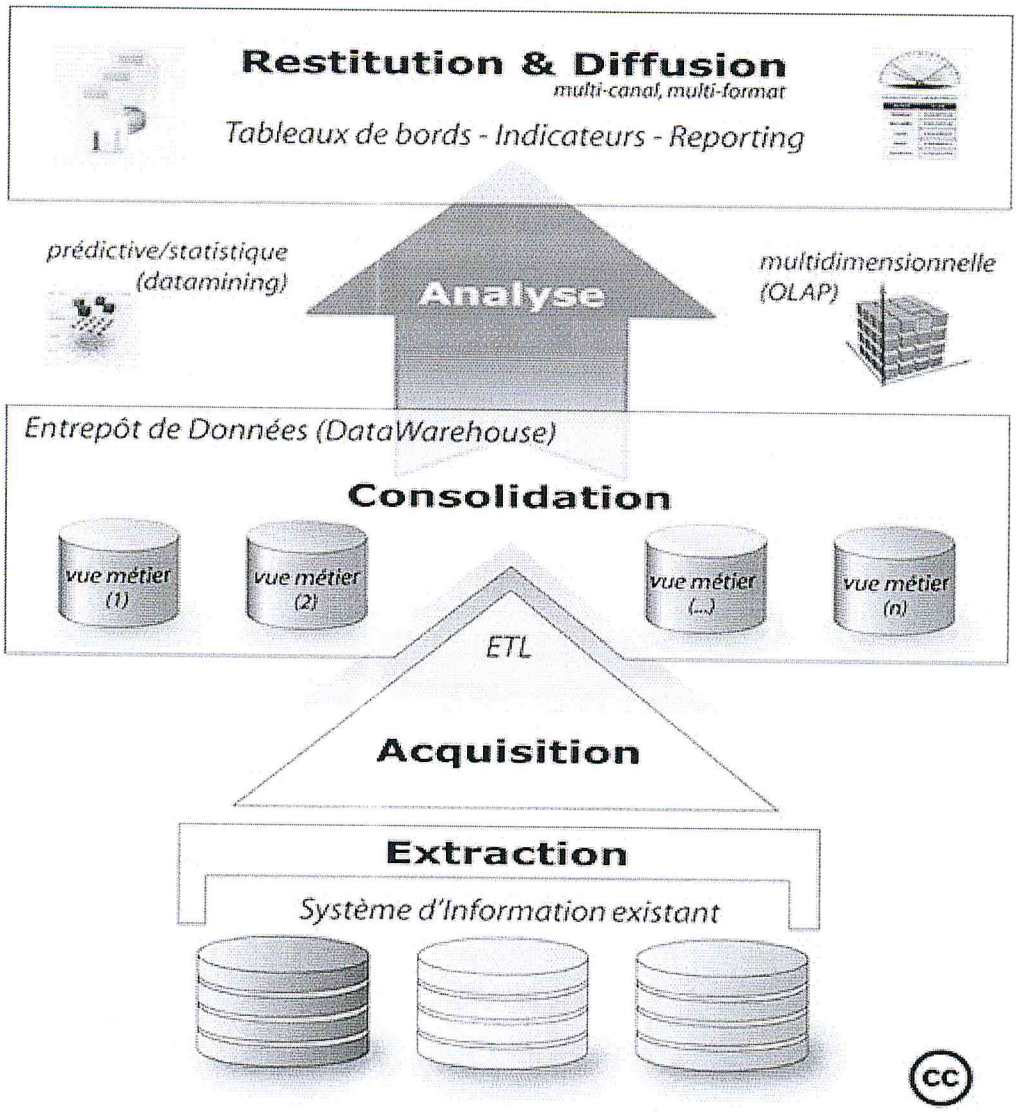


Figure 1: La chaine décisionnelle [W09]

1.4.1 Extraction

L'extraction s'effectue à partir de données appelées : données sources. Ces données peuvent se présenter sous différents formats. Il peut s'agir de fichiers "plats" (fichiers CSV avec séparateurs, fichiers XML, fichiers ASCII...) mais aussi de systèmes de bases de données (export de base MySQL, PostgreSQL, DB2, ORACLE...). Ces sources de données sont donc en général hétérogènes c'est pourquoi il va falloir passer par une phase dites acquisition pour pouvoir les manipuler avant de les stocker dans notre système d'aide à la décision.

1.4.2 Acquisition

C'est à ce niveau qu'apparait la première couche logicielle de l'environnement décisionnel à savoir l'ETL. Cette couche offre des fonctions d'extraction de données issues de différents systèmes (internes ou externes), de transformation de ces données

(homogénéisation, filtrage, calcul) et de leur chargement dans le DW (entrepôt de données). Elle garantit la délocalisation de la charge de calcul et une meilleure disponibilité des sources.

1.4.3 Consolidation

Permet de stocker les données dans un entrepôt appelé : Datawarehouse. Cet entrepôt contient les données orientées métier, non volatiles (datées), historisées et documentées. Cette structure de données est volontairement généralement dénormalisée pour pouvoir optimiser les temps de réponses lorsque l'on fait des analyses de type OLAP qui se réfère à une base de données multidimensionnelle (aussi appelée cube ou hypercube). Elle est constituée de dimensions ou axes d'analyse (l'axe temporel ou géographie sont des exemples courant) et de faits ou indicateurs (tels que le chiffre d'affaires). Un élément important vient du fait que les données stockées dans le DW ne doivent plus changer une fois à l'intérieur. Ce sont des données consolidées et figées qui vont nous permettre de faire toute sorte d'analyses et statistiques.

1.4.4 Analyse et restitution

On distingue à ce niveau plusieurs types d'outils différents :

- Les outils de reporting et de requêtes
- Les outils d'analyse
- La phase de Datamining

Les outils de reporting et de requêtes permettent la mise à disposition de rapports périodiques, pré-formatés et paramétrables par les opérationnels. Ils offrent une couche d'abstraction orientée métier pour faciliter la création de rapports par les utilisateurs eux-mêmes en interrogeant le DW (entrepôt de données) grâce à des analyses croisées. Ils permettent également la production de tableaux de bord avec des indicateurs de haut niveau pour les managers, synthétisant différents critères de performance.

Les outils d'analyse OLAP permettent de traiter des données et de les afficher sous forme de cubes multidimensionnels et de naviguer dans les différentes dimensions. Cet agencement des données permet d'obtenir immédiatement plusieurs représentations d'un même résultat, en une seule requête sous une approche descendante des niveaux agrégés vers les niveaux détaillés.

Les outils de Datamining offrent une analyse plus poussée des données historisées permettant de découvrir des connaissances cachées dans les données comme la détection de corrélations et de tendances, l'établissement de typologies et de segmentations ou encore des prévisions. Le Datamining est basé sur des algorithmes statistiques et mathématiques, et sur des hypothèses métier.

2. Entrepôt de données

2.1 Définition

« Le DataWareHouse est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision » [03].

L'entrepôt de données est un environnement informationnel qui :

- Fournit une vue intégrée et totale de l'entreprise.

- Rend l'information courante et historique de l'entreprise facilement disponible pour la prise de décision.

- Rend les transactions aide à la décision possible sans gêner les systèmes opérationnels.

- Rend les informations de l'organisation cohérente.

- Présente une source d'information stratégique flexible et interactive. [11]

Voici la définition du grand dictionnaire :

«Un entrepôt de données est une structure informatique dans laquelle est centralisé un volume important de données consolidées à partir des diverses bases de données internes et externes d'une entreprise, et qui est conçue pour offrir un accès rapide à l'information stratégique nécessaire à la prise de décision ».

2.2 Type des données d'entrepôt

2.2.1 Données intégrées

Les sont intégrées car provenant des différents systèmes opérationnels hétérogènes ou d'origines diverses de l'organisation. Ces données doivent être standardisées, épurées,

unifiées et homogénéisées pour assurer la cohérence et l'intégrité de la connaissance de l'entreprise. [04]

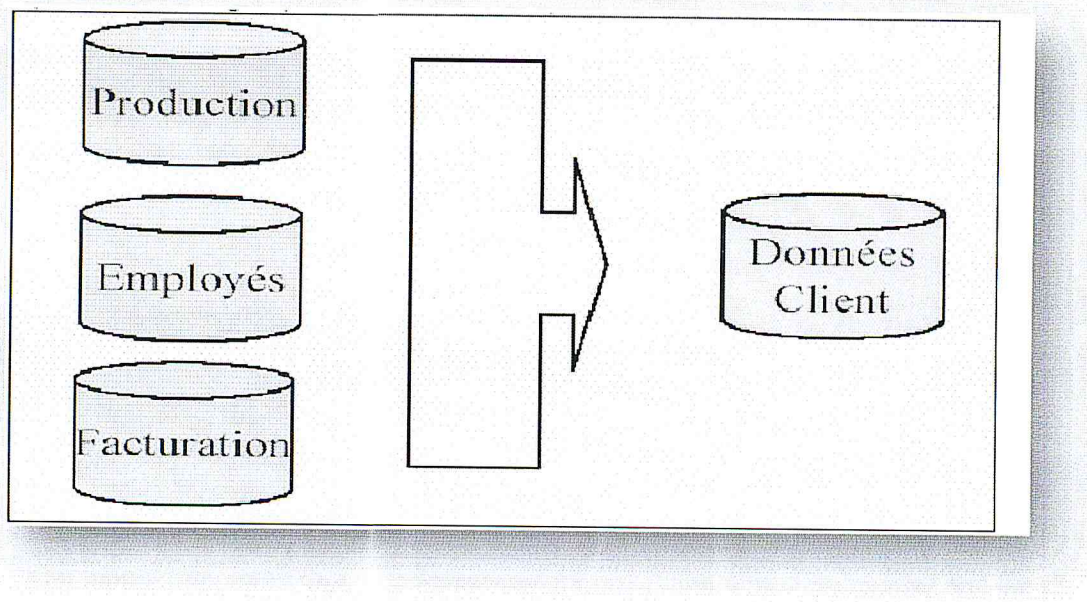


Figure 2: Données intégrées [W01]

2.2.2 Données orienté sujet

Données orientée sujet ou business et organisée par thème afin de permettre la réalisation d'analyses autour des sujets primordiaux et des métiers de l'entreprise. Par cette organisation on peut ainsi passer d'une vision verticale de l'entreprise à une vision transversale beaucoup plus riche en information. [04]

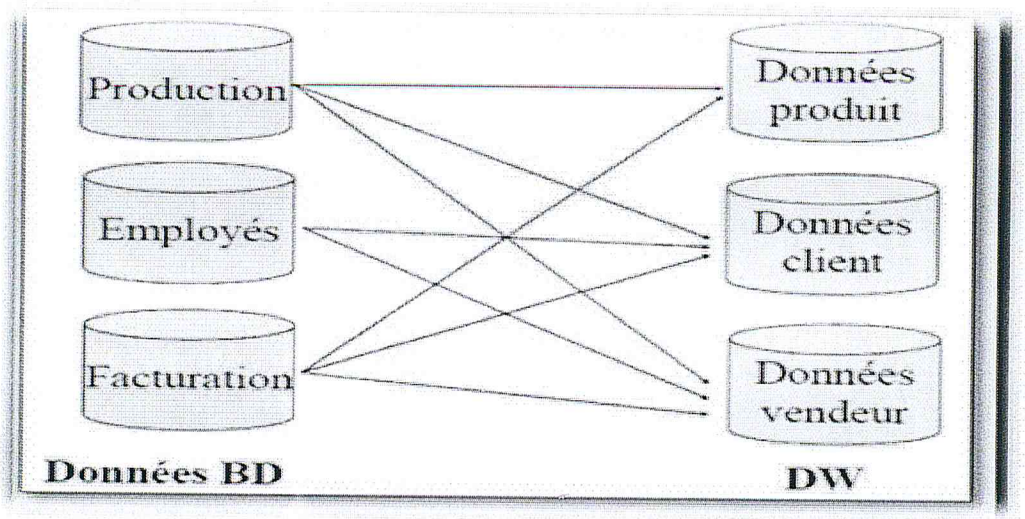


Figure 3: Données orienté sujet [W01]

2.2.3 Données historisées

Ce sont des données archivée et datée pour le suivi des évolutions des valeurs des indicateurs à analyser. Les informations stockées au sein du DW ne doivent pas disparaître. Elles deviennent, de ce fait, partie prenante de l'historique de l'entreprise. [04]

2.2.4 Données non volatiles

Un entrepôt de données doit conserver la traçabilité des informations et des décisions prises. Les données ne sont ni modifiées ni supprimées. Une requête émise sur les mêmes données à plusieurs mois d'intervalles doit donner le même résultat.
Un entrepôt de données définis donc à la fois un ensemble de données et un ensemble d'outils. Il s'agit de données destinées aux décideurs, qui sont souvent une copie des données de production avec une valeur ajoutée (orientés sujets, agrégées, historisées). Et c'est un ensemble d'outils permettant de regrouper les données des différentes sources, de les nettoyer et de les intégrer, ainsi que d'y accéder de différentes manières (requêtes, rapport, analyse, datamining). [W01]

2.3 Les classes de données

Un entrepôt de données peut se structurer en quatre classes de données organisées selon un axe historique et un axe de synthèse. [15]

2.3.1 Les données agrégées

Les données agrégées correspondent à des éléments d'analyse représentant les besoins des utilisateurs. Elles constituent déjà un résultat d'analyse et une synthèse de l'information contenue dans le système décisionnel, et doivent être facilement accessibles et compréhensibles.

2.3.2 Les données détaillées

Les données détaillées reflètent les événements les plus récents. Les intégrations régulières des données issues des systèmes de production vont habituellement être réalisées à ce niveau.

2.3.3 Les métadonnées

Les métadonnées constituent l'ensemble des données qui décrivent des règles ou processus attachés à d'autres données. Ces dernières constituent la finalité du système d'information.

2.3.4 Les données historisées

Chaque nouvelle insertion de données provenant du système de production ne détruit pas les anciennes valeurs, mais créée une nouvelle occurrence de la donnée.

2.4 Magasin de données (datamarts)

«Le magasin de données est issu d'un flux de données provenant du DW. Contrairement à ce dernier qui présente le détail des données pour toute l'entreprise, il a pour vocation de présenter la donnée de manière spécialisée, agrégée et regroupée fonctionnellement » *[03]*

Un entrepôt de données peut prendre la forme d'un DW, d'un magasin de données ou d'un comptoir de données (data mart). En général, le DW globalise toutes les données applicatives de l'entreprise, tandis que les magasins de données sont des sous-ensembles d'informations concernant un domaine particulier de l'entreprise [06].

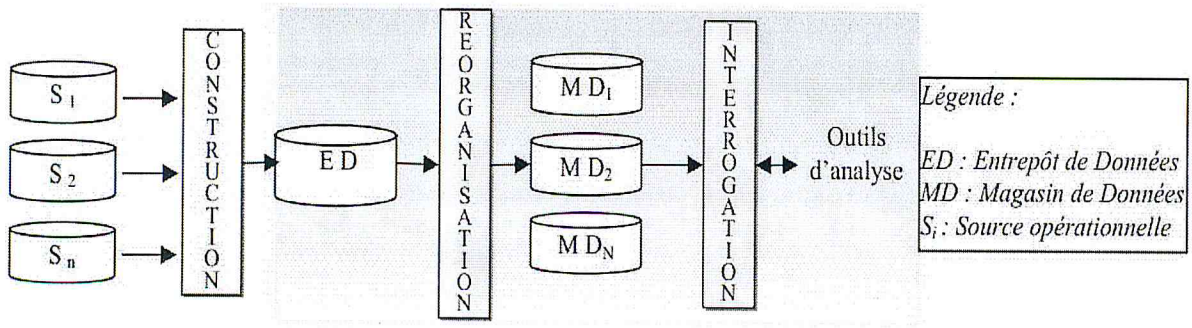


Figure 4 : Entrepôt et magasins de données [07]

2.5 Modélisation de données

2.5.1 Modélisation par sujet

La modélisation par sujet est une technique de conception logique qui vise à organiser et classifier les informations des bases légataires en données classées par sujet fonctionnel. Elle est basée sur la modélisation " Entité/Relation " et est préliminaire à la modélisation dimensionnelle. Chaque sujet correspond à une table gérée au sein de l'entrepôt. Il faut isoler

les données stratégiques, déterminer les informations de détails nécessaires (profondeur, granularité) et conserver les métadonnées [15].

2.5.2 Modélisation dimensionnelle

Le modèle dimensionnel répond aux lacunes des modèles transactionnels. Il vise à présenter les données sous une forme intuitive dont l'objectif est de se rapprocher de la manière dont les décideurs perçoivent les données d'analyse [08] [05]. Ce modèle propose de visualiser les données représentant les sujets d'analyse comme des points dans un espace à plusieurs dimensions formant les différents axes d'analyse [09].

Dans la conception, Le modèle multidimensionnelle est basé sur les deux concepts suivants [05] :

- **Fait :** Un fait est un centre d'intérêt décisionnel. Il regroupe un ensemble d'attributs numériques représentant les mesures d'activité. Il représente un sujet d'analyse dans une application décisionnelle. Supposons, par exemple, que nous souhaitons analyser les performances des agences dans une société de location de véhicules. Dans un schéma dimensionnel, ce besoin est modélisé par le fait Location. Afin de calculer la performance des agences, nous définissons les mesures d'activités montant et durée des locations dans le fait Location.
- **Dimension :** les dimensions ou axes d'analyse sont des entités complémentaires qui décrivent la table de faits. Elles contiennent, autant que possible, des attributs sous forme de descriptions textuelles permettant de qualifier ou d'expliquer l'activité. Des attributs de dimensions, nombreux, permettent de varier les possibilités d'analyse (par tranches ou en dés). Ces attributs rendent utilisables et intelligible les données de l'entrepôt de données. Ils établissent, en quelque sorte une interface homme/entrepôt de données.

2.5.2.1 Types de base du modèle dimensionnel

a. Schémas en étoile

Il consiste en une grande table de faits et un cercle d'autres tables qui contient les éléments descriptifs du fait, appelées « dimensions ». Quand illustré, le modèle ressemble à une étoile, c'est d'ailleurs l'origine du terme « En étoile ». [W06]

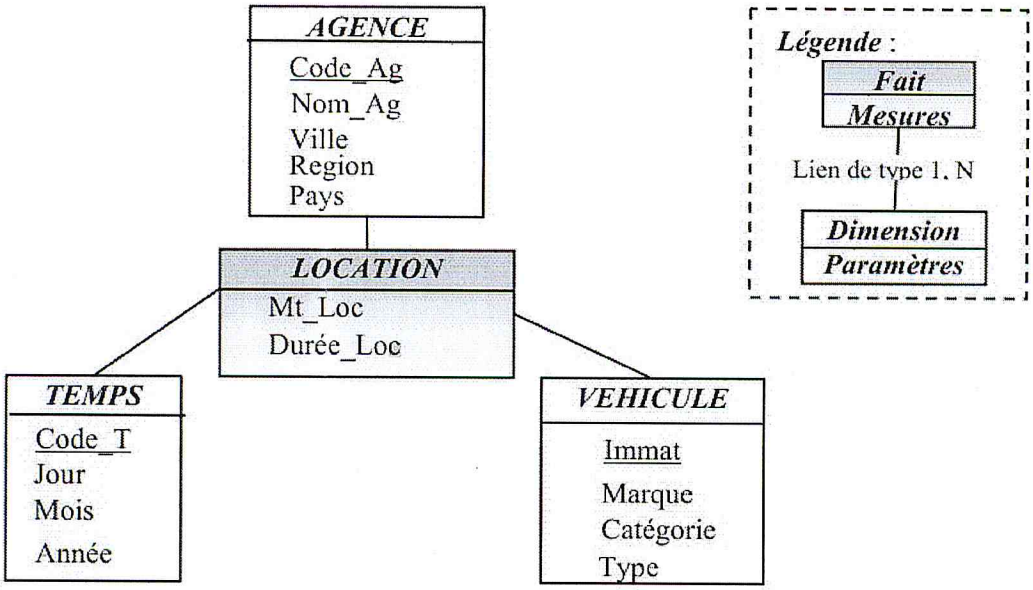


Figure 5 : schéma en étoile [05]

b. Schémas en flocons de neige

Le schéma en flocon permet d'expliciter la structure des hiérarchies en normalisant les dimensions. Par contre, il nécessite de définir des jointures entre les différentes tables de niveaux hiérarchiques pour interroger les données d'une dimension. Le temps de réponse aux requêtes mettant en jeu un volume très important de données devient alors inacceptable dans un contexte d'analyse interactive [05]

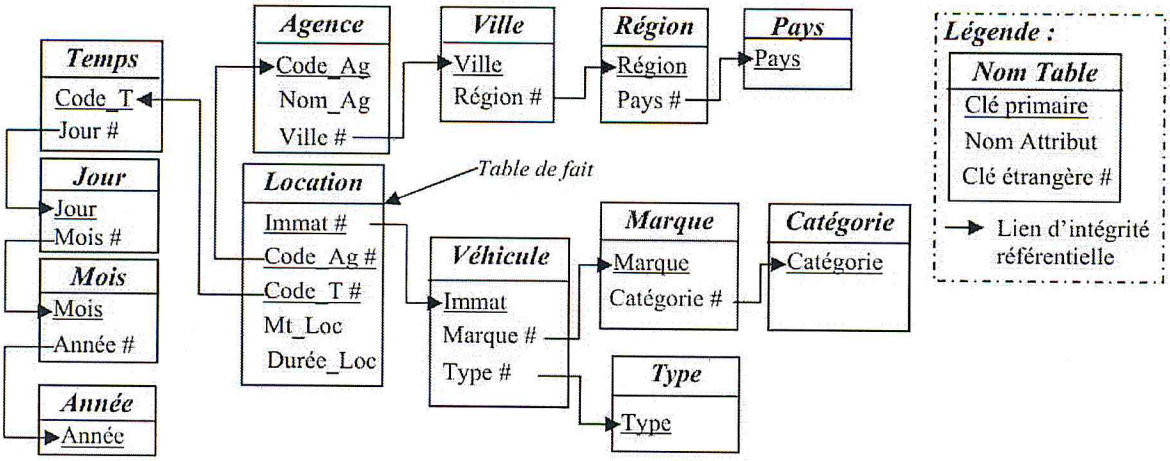


Figure 6 : Schémas en flocons de neige [05]

c. Schémas en constellation

Ce schéma est une extension du schéma en étoile (Voir Figure 7). Il consiste à fusionner plusieurs schémas en étoile qui utilisent des dimensions communes. Un schéma en constellation comprend donc plusieurs faits reliés à un ensemble de dimensions qui peuvent être partagées.

Ce schéma présente l'avantage de pouvoir corréler les sujets d'analyse tels que la comparaison des montants des locations réalisées dans les différentes agences par rapport aux chiffres d'affaires réalisés par son personnel. En outre, le partage des dimensions par plusieurs faits permet d'éviter de les définir plusieurs fois [10].

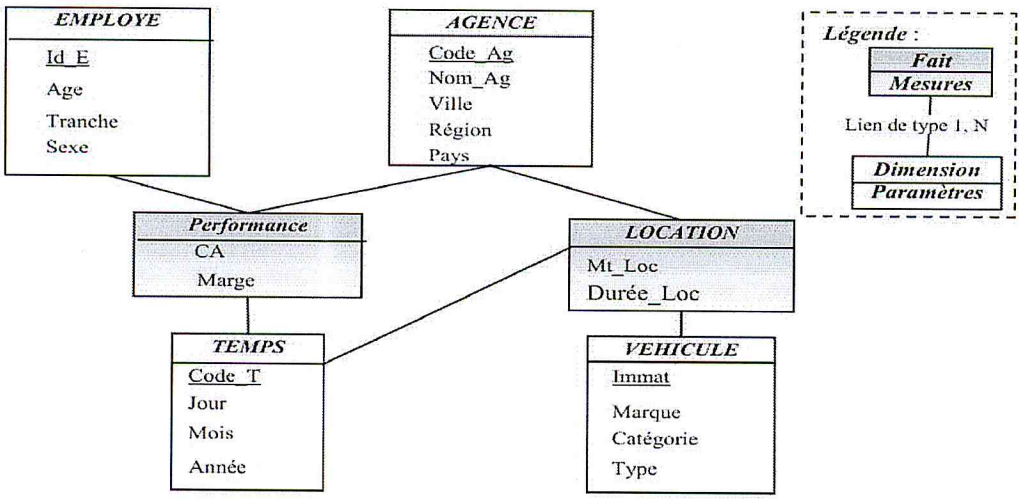


Figure 7 : Schémas en constellation [05]

2.6 Manipulation dimensionnelle

Les données dimensionnelles sont représentées au travers d'un cube regroupant à la fois la structure et les valeurs des données (voir Figure 8). Chaque case dans le cube présente les valeurs des mesures d'un fait (par exemple les montants des locations sont représentées à l'intersection des dimensions Agence, Véhicule et Temps). Chaque arête du cube, représentant une dimension, est composée des valeurs d'un paramètre de la dimension considérée.

La Figure 8 présente le cube analysant les mesures du fait Location en fonction des paramètres Année de la dimension Temps, Marque de la dimension Véhicule et Ville de la dimension Agence.

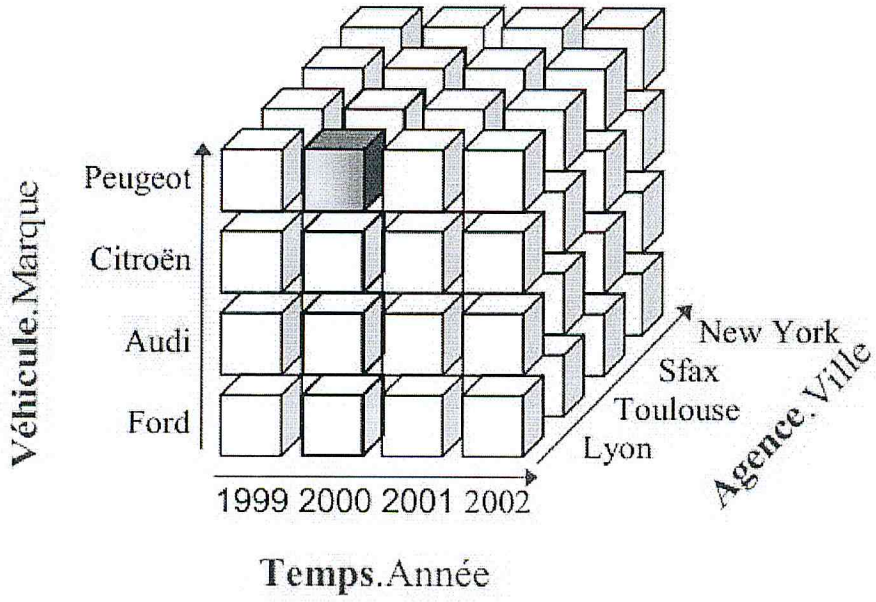


Figure 8: Exemple de cube dimensionnel [10]

Une nouvelle génération d'opérateurs algébriques basés sur le concept de cube a vu le jour. La représentation dimensionnelle fait appel à des opérateurs spécifiques qui faciliteront l'analyse et la visualisation des cubes dimensionnels [08].

2.6.1 Opérateur de rotation

Permet d'avoir accès aux différentes vues de données : c'est le fait d'inverser les axes visualisés du cube. Un cube de n dimensions possède n * (n – 1) vues possibles.

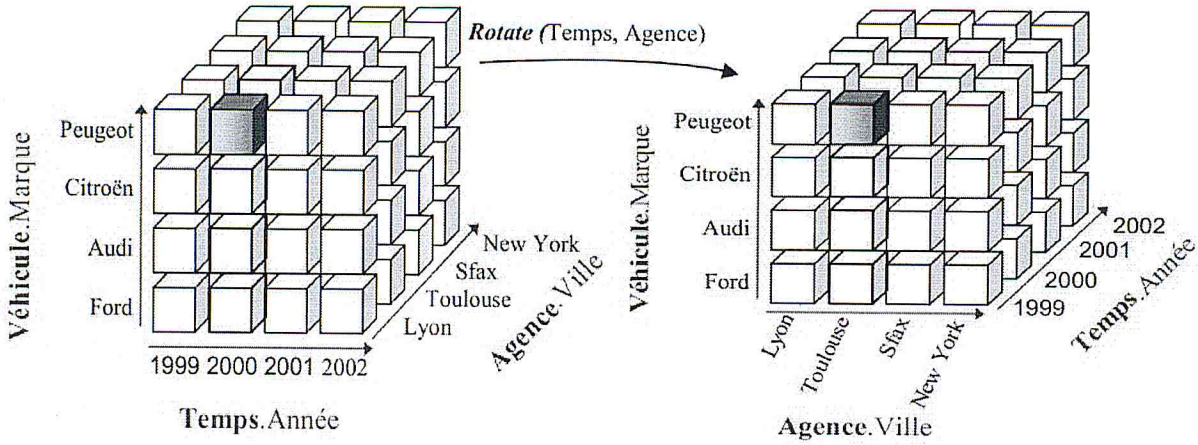


Figure 9: Rotation des dimensions agence et véhicule [10]

Cette rotation du cube, nous permet de visualiser les locations en fonction des marques de véhicules et des villes. Les locations en fonction des années passent au plan latéral.

2.6.2 Opérateur de restriction

Permettent de restreindre les valeurs dans le cube. Slice est appliqué sur les dimensions tel que restreindre l'analyse des ventes aux villes de Toulouse et de Lyon (Figure 10). Dice sert à restreindre les données des faits.

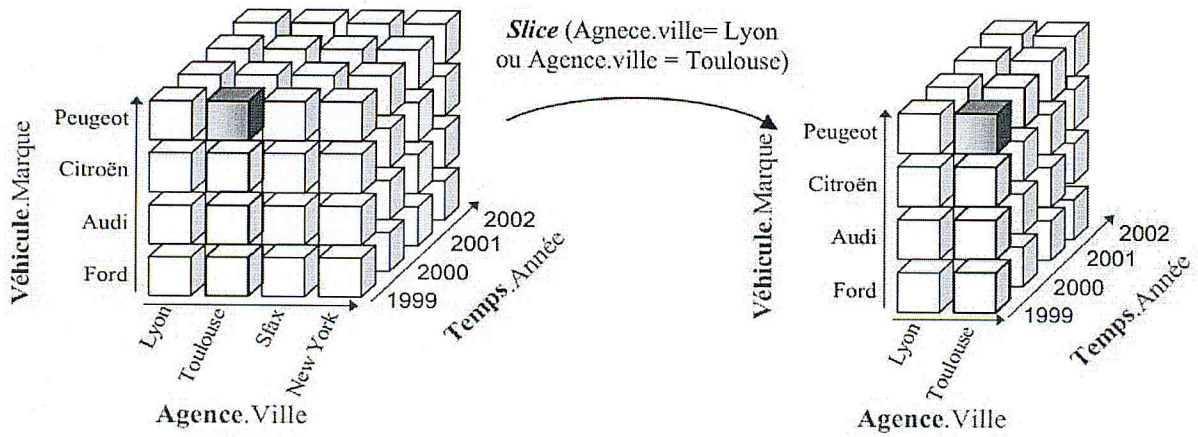


Figure 10: L'operateur de restriction slice [10]

2.6.3 Opérateur de forage vers le haut et de forage vers le bas

Permettent soit de généraliser l'analyse, soit de l'affiner en modifiant le paramètre utilisé pour définir les valeurs d'une arête du cube. En effet, les dimensions sont associées à des hiérarchies ; ces deux opérateurs permettent, respectivement, de « monter » ou de « descendre » dans une hiérarchie.

Dans l'exemple de la Figure 9, nous présentons une opération de forage vers le bas à partir du paramètre Ville vers le paramètre Code_Ag.

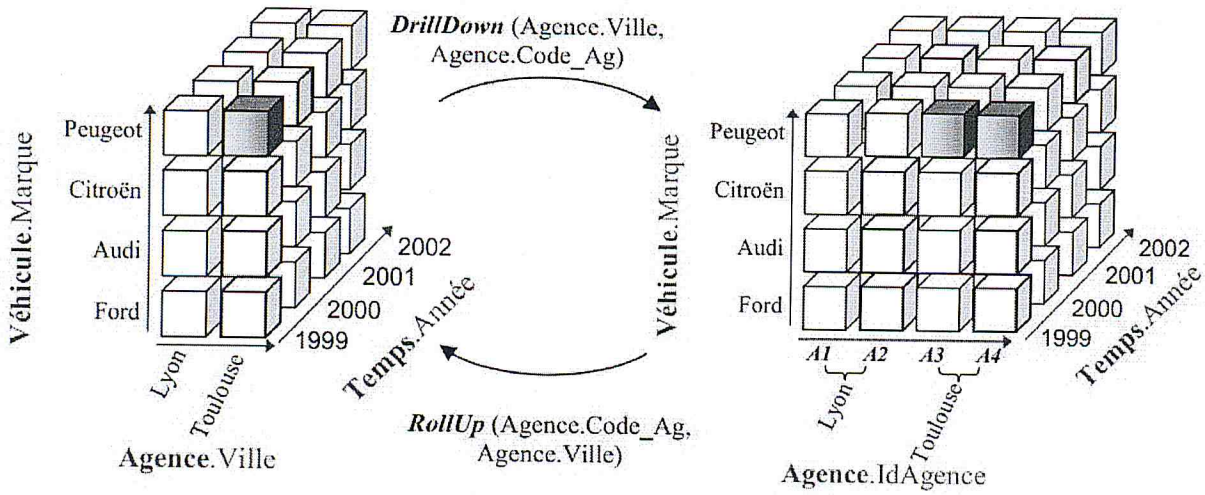


Figure 11: Les operateurs de forage [10]

2.7 Les objectifs d'un entrepôt de données

- ✓ Accès aux informations de l'entreprise.
- ✓ Les informations de l'entreprise sont cohérentes.
- ✓ Une vue consistante, globale et unifiée de l'entreprise.
- ✓ Les données publiées sont entreposées pour la consultation rapide de l'entreprise.
- ✓ Qualité de l'information d'un entrepôt de données.
- ✓ Les outils de présentation d'informations font partie de l'entrepôt de données
- ✓ Supporte des applications de type informationnel comme, les applications d'aide à la décision.

3 Conclusion

A partir de ce que nous avons vu dans ce chapitre, nous constatons que les flux d'alimentation de données dans la chaine décisionnel sont réalisés avec des outils ETL (Extraction-Transformation-Loading) qui assurent le transfert et la mise en forme des données entre les systèmes sources et l'entrepôt de données. De plus cette phase est très importante dans la chaine décisionnelle, il se doit répondre à des normes de performance fortement exigeantes. Par la suite nous allons étudier le processus ETL de manier approfondie dans le chapitre suivant.

Chapitre II

Processus ETL

Chapitre II : Processus ETL

À présent que nous avons survolé les grands principes et concepts de l'informatique décisionnelle, nous pouvons nous attarder sur le processus ETL qu'est le vif du sujet. À l'origine, le principe est simple: il s'agit d'alimenter les entrepôts de données. Maintenant, les ETL ce sont largement diversifiés et permettent d'effectuer de nombreuses opérations que nous verrons par la suite. Concrètement, on dispose de sources (souvent hétérogènes) à partir desquelles sont extraites des données pour alimenter un entrepôt pour des fins d'analyse. Les sources peuvent aussi bien être des bases de données (de n'importe quel SGBD), des fichiers (CSV, XML, Excel) et voire d'autres formats (annuaires LDAP, Web services). Les ETL s'occupent de transformer ces sources, via de nombreux composants, en une ou plusieurs cibles qui peuvent être, là aussi, de n'importe quels formats.

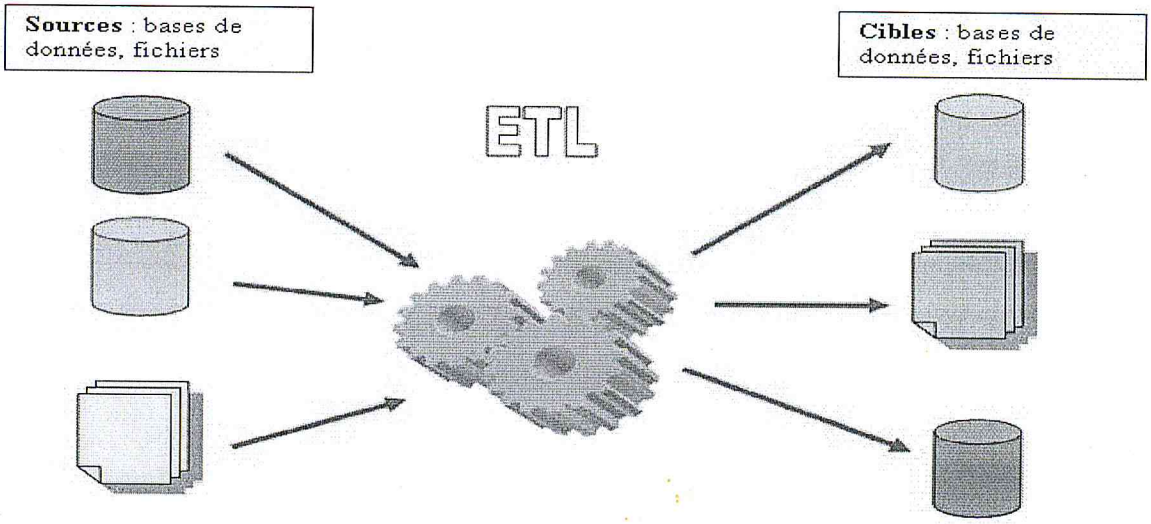


Figure 12: Processus ETL [13]

1 Extract-Transform-Load

Comme expliqué précédemment, ETL signifie Extracting-Transforming-Loading (en français Extraction-Transformation-Chargement). Ce sont les trois étapes que doit impérativement implémenter un outil ETL. Effectuées dans l'ordre, elles forment un traitement, une tâche ou un scénario (selon les diverses appellations des logiciels). Nous allons à présent les détailler.

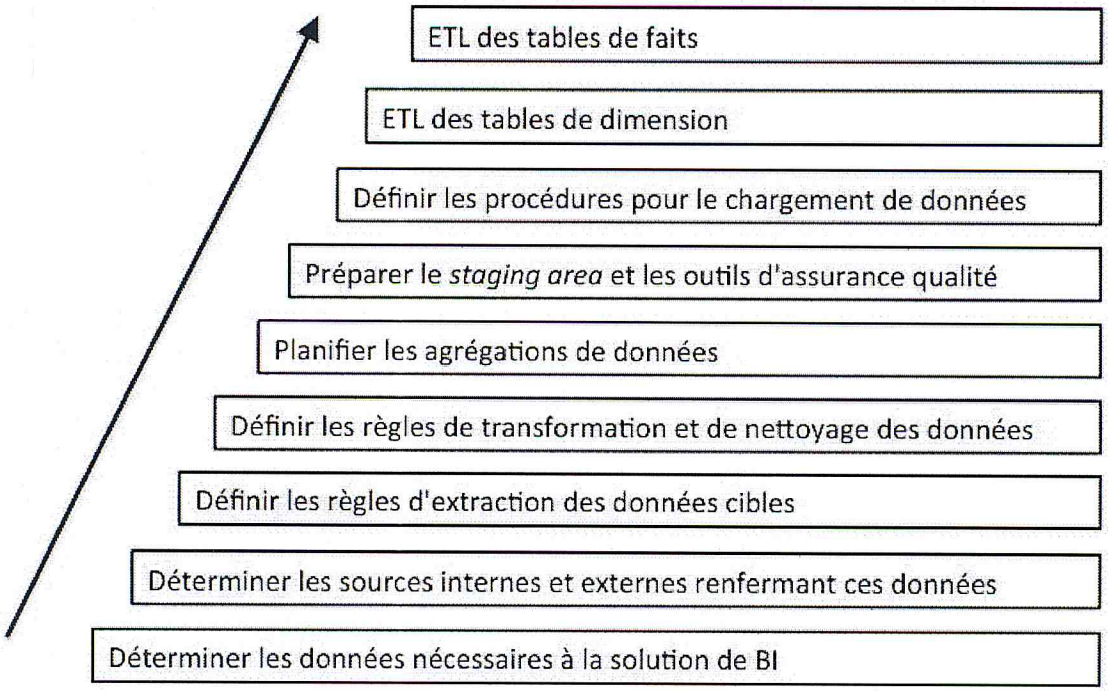


Figure 13: Etapes d'ETL [13]

1.1 Extraction

a. Sélection des données sources

La première étape concerne l'extraction des données qui sont la plupart du temps hétérogènes. Cela signifie qu'elles peuvent provenir de SGBD (MySQL, Oracle, SQL Server, etc.), de fichiers plats (Txt, Excel, XML, etc.), d'ERP (Enterprise Ressource Planning), de bases hiérarchiques (les ancêtres des SGBD) ou d'autres applications spécifiques. On peut déjà remarquer le premier obstacle aux ETL : la multitude de formats sources possibles à gérer.

Ainsi que les données parfois sont diffuses et complexes, cela signifie qu'elles peuvent provenir de différents environnements matériels et différents réseaux interconnectés ou non avec différents modèles logiques et physiques principalement orientés vers les traitements transactionnels.

Pour sélectionner les données utiles à partir des BD de production il faut suivre les étapes suivant [13] :

- ✓ Énumérer les items cibles (métriques et attributs de dimension) nécessaires à l'entrepôt de données.

- ✓ Pour chaque item cible, trouver la source et l'item correspondant de cette source.
- ✓ Si plusieurs sources sont trouvées, choisir la plus pertinente.
- ✓ Si l'item cible exige des données de plusieurs sources, former des règles de consolidation.
- ✓ Si l'item source referme plusieurs items cibles (ex : un seul champ pour le nom et l'adresse du client), définir des règles de découpage.
- ✓ Inspecter les sources pour des valeurs manquantes.

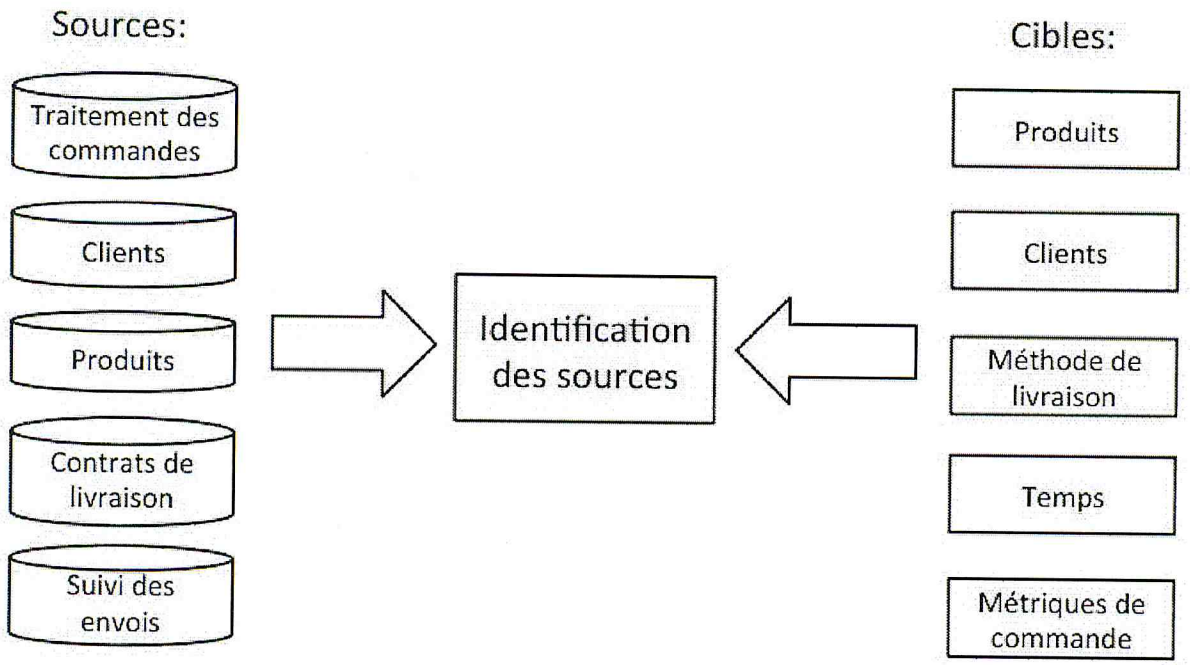


Figure 14: Exemple d'identification des sources [13]

Il est aussi important, lorsque l'on extrait des données, de pouvoir les analyser. Il faut donc connaître les propriétés de celles-ci : savoir si par exemple cette donnée est de type entier ou chaîne de caractères et quelle est sa taille maximale. Cela peut paraître simple lorsque la source provient d'un SGBD mais s'avère plus complexe lorsqu'elle provient d'un fichier plat. Il faut aussi pouvoir reconnaître les clés primaires et étrangères permettant respectivement d'identifier une table de façon unique, et de garantir l'intégrité des données.

L'extraction peut aussi s'occuper de vérifier les erreurs des sources. Par exemple, il est possible qu'une personne fasse une faute de frappe en écrivant "Canuda" au lieu de "Canada" alors l'outil ETL doit détecter cette erreur et la corriger.

L'étape d'extraction est donc très importante. Elle doit être performante et complète pour pouvoir disposer d'un bon outil ETL.

b. Techniques d'extraction

L'étape d'extraction doit être conçue d'une manière qui n'affecte pas négativement à la performance et le temps de réponse ou tout type de blocage de système d'extraction.

b.1 Extraction complète

Certains systèmes ne sont pas en mesure d'identifier lesquels des données a été changé du tout, donc un extrait complet est la seule façon d'obtenir les données du système. L'extrait complet nécessite en gardant une copie du dernier extrait dans le même format afin d'être en mesure d'identifier les changements. Extrait complet gère suppressions ainsi. Donc il est employé dans deux situations [13] :

- chargement initial des données.
- Rafraichissement complet des données (ex : modification d'une source).

Note : Peut être très couteuse en temps (ex : plusieurs jours).

b.2 Extraction incrémentale

Capture uniquement les données qui ont changées ou ont été ajoutées depuis la dernière extraction, cette technique peut être faite de deux façons [13]:

b.2.1 Extraction temps-réel

Si le système source est capable de fournir une notification que le document a été modifié et décrire le changement, c'est la meilleure façon d'obtenir les données, avec cette technique l'extraction s'effectue au moment où les transactions surviennent dans les systèmes sources.

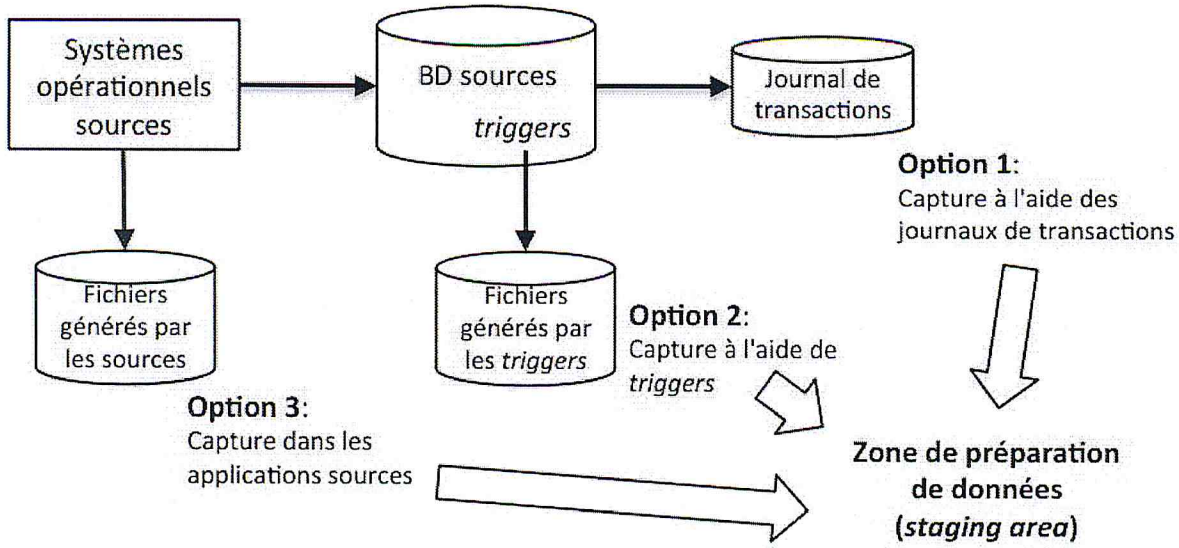


Figure 15: Extraction temps réel [13]

Option 1: Capture à l'aide du journal de transactions

- Utilise les logs de transactions de la BD servant à la récupération en cas de panne.
- Aucune modification n'est requise à la BD ou aux sources.
- Doit être fait avant le rafraîchissement périodique du journal.
- Pas possible avec les systèmes legacy ou les sources à base de fichiers (il faut une BD journalisée).

Option 2: Capture à l'aide de triggers

- Des procédures déclenchées (triggers) sont définies dans pour recopier les données à extraire dans un fichier de sortie.
- Meilleur contrôle de la capture d'évènements.
- Exige de modifier les BD sources.
- Pas possible avec les systèmes legacy ou les sources à base de fichiers.

Option 3: Capture à l'aide des applications sources

- Les applications sources sont modifiées pour écrire chaque ajout et modification de données dans un fichier d'extraction.
- Exige des modifications aux applications existantes.
- Entraîne des coûts additionnels de développement et de maintenance.

- Peut être employé sur des systèmes legacy et les systèmes à base de fichiers.

b.2.2 Extraction différée

Extrait tous les changements survenus durant une période donnée (ex: heure, jour, semaine, mois) [13].

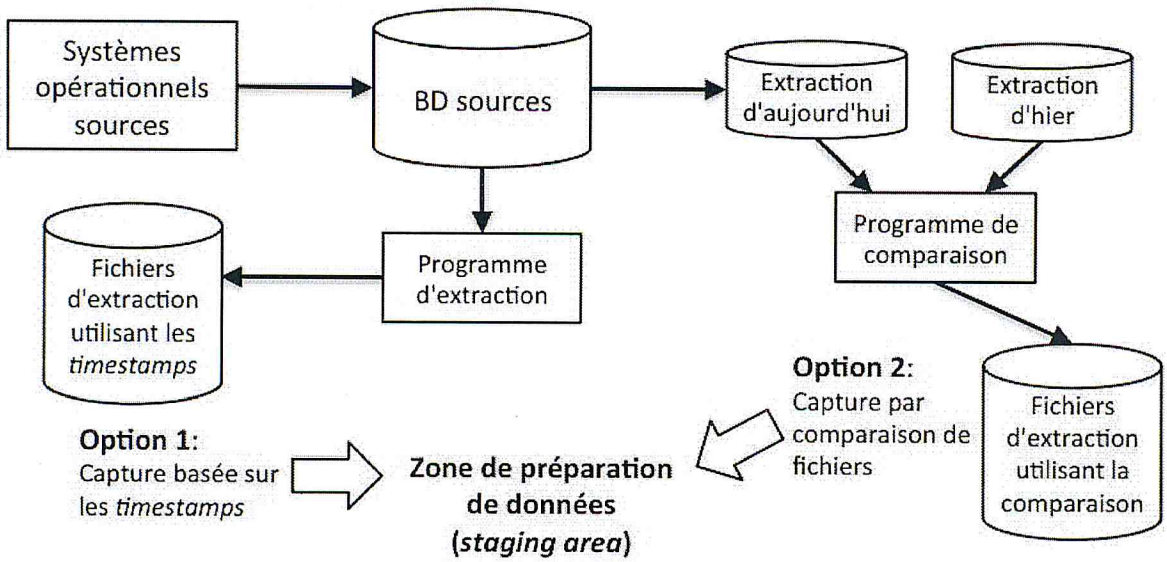


Figure 16: Extraction temps réel [13]

Option 1 : Capture basée sur les timestamps

- Une estampille (timestamp) d'écriture est ajoutée à chaque ligne des systèmes sources.
- L'extraction se fait uniquement sur les données dont le timestamp est plus récent que la dernière extraction.
- Fonctionne avec les systèmes legacy et les fichiers plats, mais peut exiger des modifications aux systèmes sources.
- Gestion compliquée des suppressions.

Option 2 : Capture par comparaison de fichiers

- Compare deux snapshots (captures) successifs des données sources.
- Extrait seulement les différences (ajouts, modifications, suppressions) entre les deux snapshots.
- Peut être employé sur des systèmes legacy et les systèmes à base de fichiers, sans aucune modification.
- Exige de conserver une copie de l'état des données sources.

- Approche relativement couteuse

Lors de l'utilisation d'extraits incrémentielle ou complète, la fréquence d'extraction est extrêmement importante. Notamment des extraits complets, les volumes de données peuvent être dans des dizaines de giga-octets ou des péta-octets.

c. La Complexité de l'extraction

L'étape de l'extraction doit être effectuée le plus rapidement possible en exploitant au minimum les ressources du système C'est pour cette raison que l'on effectue rarement des transformations lors de l'extraction d'une part. D'autre part, on essaye au maximum d'extraire seulement les données utiles (Mise à jour ou ajoutée après la dernière extraction) et pour ce faire on pourrait s'entendre avec le responsable du système source pour ajouter soit un flag ou encore des dates dans chacune des tables extraites, au moins deux dates : Date de création de l'enregistrement dans la table et la date de mise à jour (En général la plupart des systèmes sources disposent de ces deux dates) . Par ailleurs pour ne pas perdre des données suites à des problèmes d'extraction, il est important de s'assurer que le système source ne purge pas les données avant que l'entrepôt ne les aies extraits.

La complexité de l'extraction est liée avec ces processus [13] :

- Avoir des informations complètes et détaillées sur chaque structure de sources impliquées dans l'extraction.
- Connaitre la fenêtre de temps de chaque source, durant laquelle peut se faire l'extraction.
- Déterminer des mécanismes permettant de capturer les changements dans les données des sources pertinentes.

1.2 Transformation

Cette seconde étape a pour objectif la transformation des données. Elle est bien évidemment indispensable si l'on veut obtenir des cibles différentes des sources.

C'est cette étape qui va permettre de joindre les différentes sources selon les clés précédemment spécifiées. Elle va aussi permettre de filtrer les données. Le filtrage est bien différent de l'extraction puisque l'on filtre selon des critères à définir.

Une partie importante de l'étape de transformation est de pouvoir effectuer des calculs.

Ils peuvent être simples comme une addition ou multiplication, mais peuvent être aussi plus complexes. Disposer d'un outil ETL proposant de nombreuses opérations par défaut est donc un plus.

a. Types de Transformations

Voici l'ensemble de types de transformation [13]:

1. **Révision de format :** Cette tâche vise à changer le type ou la longueur de champs individuels.
2. **Pré-calcul des valeurs dérivées :** cette tâche à pour but de calculer des nouveaux champs à partir des données source par exemple : Profit calculé à partir de ventes et coût.
3. **Fusion de plusieurs champs :** comprend la fusion et la combinaison champs provenant de plusieurs sources des données et produits une nouvelle entité par exemple : pour les informations d'un produit, il fait la combinaison de ces champs source 1 : code et description, source 2 : types de forfaits, source 3 : coût.
4. **Récapitulation:** Les valeurs sont résumées pour obtenir des chiffres totaux qui sont ensuite calculées et stockées à plusieurs niveaux dans les tables de faits multidimensionnels.
5. **Dérivation** : - ce type de transformation vise à créée des nouvelles données à partir de données de sources existantes (détaillé) pendant ce processus par des calculs, par exemples la dérivation peuvent être: le calcul de l'âge d'un client en fonction de leur date de naissance et l'année en cours.
6. **Décodage de champs** : c'est la consolidation les données de source multiples par exemple : ['homme','femme'] vers ['M','F'] et aussi pour traduire les valeurs cryptiques par exemple : 'AC','IN','SU' pour les statuts actif, inactif et suspendu.
7. **Agrégations** : La transformation doit aussi s'occuper des différentes agrégations : la somme ou la moyenne d'un champs par d'autre champs par exemple : Ventes par produit par semaine par région.
8. **Conversion de dates :** comprend la conversion de n'importe format de date vers une autre format par exemple : 'Z4 FEB 2011' vers '24/02/2011'.

9. **Déduplication :** par fois on rencontre des duplications dans les bases de données et ils peuvent produire des problèmes par exemple : plusieurs enregistrements pour un même client. Donc cette tache consiste à éliminer cette duplication.

10. Clé de substitution

L'outil ETL peut également permettre de générer des clés de substitution (surrogate key) pour cette étape de transformation. Chaque ligne d'un ensemble de données peut posséder une clé primaire et une autre dite de substitution. La différence majeure entre ces deux clés dépend de la donnée (si elle est actuelle ou temporelle). Une base actuelle va stocker uniquement les données actuelles valides, alors qu'une base temporelle va s'occuper en plus des données précédentes. Par exemple, l'employé Jean Dupont n'aura qu'une entrée dans une base actuelle mais en aura deux dans une base temporelle (une quand il était employé de 1988 à 1992 et une autre de 2004 à 2008). Les deux clés primaires seront différentes (donc uniques) alors que les clés de substitution seront les mêmes (non uniques). Pouvoir générer ces clés accroît les performances de l'outil ETL dans la recherche des données avec une meilleure indexation. Ainsi l'outil ETL doit pouvoir permettre de garder un historique des changements et aussi obtenir une certaine indépendance des systèmes sources (c'est lui qui gère les clés de substitution).

11. Intégration des données

L'intégration de données regroupe les processus par lesquels les données provenant de différentes parties du système d'information sont déplacées, combinées et consolidées afin qu'elles deviennent prêtes pour le chargement dans l'entrepôt de données.

b. Problèmes de transformation

1. Problème de résolution d'entités

- Survient lorsqu'une même entité se retrouve sur différentes source, sans qu'on ait la correspondance entre ces sources, par exemple : Clients de longue date ayant un identifiant différent sur les différentes sources.
- L'intégration des données requiert de retrouver la correspondance.
- Approches basées sur des règles de résolution par exemple : les entités doivent avoir au moins N champs identiques [13].

2. Problème des sources multiples

- Survient lorsqu'une entité possède une représentation différente sur plusieurs sources.
- Approches de sélection : - Choisir la source la plus prioritaire
  - Choisir la source ayant l'information la plus récente [13].

c. Complexité de la transformation

Cette étape doit permettre d'effectuer toutes les transformations que l'on souhaite appliquer aux données sources. Il ne faut pas non plus oublier la sélection ou le découpage des colonnes, la traduction des valeurs (les différents formats de dates possibles ou encore le booléen 1 qui peut signifier M pour "Masculin"), la fusion, les lookups, la gestion des erreurs et encore de nombreuses autres fonctionnalités, donc cette étape est la plus lourde dans le processus d'ETL [13].

1.3 Chargement

La dernière étape, s'occupe de charger les données, préalablement extraites puis transformées, dans des cibles hétérogènes.

Le chargement va permettre d'insérer ou de mettre à jour les données cibles et comme dans les deux étapes précédentes, il doit aussi gérer les erreurs (une chaîne de caractère ne doit pas être insérée dans un champ fait pour les entiers).

a. Type de chargement

Type de chargement par [13] :

1. **Chargement initial**

- Se fait une seule fois lors de l'activation de l'entrepôt de données.
- Les index et contraintes d'intégrité référentielle (clé étrangères) sont normalement désactivés temporairement.

2. **Chargement incrémental**

- Se fait une fois le chargement initial complété;
- Tiens compte de la nature des changements (ex: SCD Type 1, 2 ou 3).
- Peut être fait en temps-réel ou en lot.

3. **Rafraîchissement complet:** Est employé lorsque le nombre de changements rend le chargement incrémental trop complexe par exemple : lorsque plus de 20% des enregistrements ont changé depuis le dernier chargement.

Le chargement n'est pas à négliger pour un bon outil ETL, il doit, là aussi, être complet et performant.

2 Autres fonctionnalités des outils ETL

Un outil ETL, qui se veut être complet, doit implémenter de nombreuses autres fonctionnalités. Il peut par exemple permettre de planifier les exécutions : lancer un traitement un jour précis ou à une fréquence précise selon une contrainte donnée (les possibilités peuvent être nombreuses). Une console d'administration est aussi la bienvenue avec l'enregistrement d'utilisateurs et de leurs privilèges, tout en permettant de surveiller les processus ETL en cours. Tout ceci doit être géré par un système bien sécurisé. Pour un travail collaboratif, on peut bien évidemment penser à des systèmes de contrôle de version genre CVS (Concurrent Versions System).

Pour optimiser les performances, un système ETL peut proposer de paralléliser les traitements et de coordonner les processus. Pour bien gérer les erreurs, l'outil ETL peut proposer des rapports d'erreurs, des outils de correction de bugs, la reprise après une erreur, la vérification d'un traitement avant son exécution ou encore l'affichage des statistiques d'exécutions.

Les possibilités d'un outil ETL sont donc très nombreuses et c'est bien évidemment un critère à retenir pour disposer d'un outil ETL complet [13].

3 Application des outils ETL à l'informatique décisionnelle

La plupart du temps, les ETL sont utilisés dans le domaine de l'informatique décisionnelle décrite dans la partie état de l'art. L'informatique décisionnelle est le plus souvent composée de différentes parties intimement liées, permettant d'aider à prendre une décision pour répondre aux problèmes décisionnels tels que :

- **L'intégration de données** qui alimente des entrepôts de données. C'est ici qu'interviennent les ETL.

- **La génération de rapports** qui fournit aux utilisateurs des rapports sur l'état des ventes, des stocks, du chiffre d'affaires, etc. Cette partie est gérée via des outils de Reporting qui piocheront dans un entrepôt de données alimenté par un ETL.
- **Les tableaux de bords** (ou dashboards en anglais) mettent en place de nombreux graphiques et schémas, pour observer, d'un coup d'œil, ce qui va ou qui ne va pas dans l'entreprise. Par exemple si un stock est quasiment vide, une jauge en rouge peut s'afficher pour prévenir rapidement l'utilisateur. Là encore, les données seront récupérées à partir d'un entrepôt de données.
- **L'analyse des données** permet d'aller plus en profondeur par rapport aux rapports mais aussi d'interagir et de vérifier les données selon plusieurs niveaux (années, trimestres, mois, semaines, jours par exemple). Les données seront récupérées via des cubes multidimensionnels OLAP, qui sont eux même alimentés par des entrepôts de données.
- **Le DataMining** est la partie la moins utilisée puisque c'est la plus complexe. Cette branche fait intervenir de nombreux algorithmes (touchant souvent au domaine de l'intelligence artificielle) essayant d'apporter à l'utilisateur les futures évolutions probables de son entreprise.

La majorité des outils ETL mettent à disposition des outils spécifiques pour alimenter les entrepôts de données. Les clés de substitutions présentées plus haut, l'alimentation de cubes OLAP ou encore la gestion des dimensions à évolution lente (slow changing dimension) en sont des exemples [13].

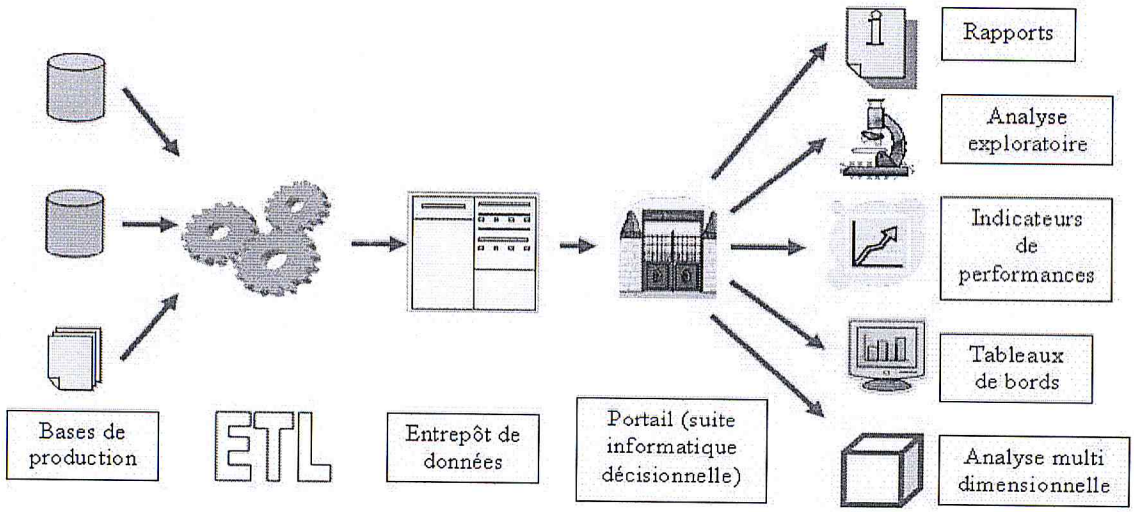


Figure 17 : Schéma explicatif de l'utilisation d'outils ETL dans l'informatique décisionnelle [13].

4 Autres applications des outils ETL

Nous avons vu précédemment l'application des ETL à des fins décisionnelles mais il ne faut pas oublier que les outils ETL peuvent servir dans d'autres applications [13].

4.1 Migration de données

Lorsqu'une entreprise souhaite passer d'une version d'une base de données ou d'une application à une autre, ou même lorsqu'elle change de système, elle doit effectuer ce que l'on appelle une migration de données. Toutes les données existantes doivent être transférées dans un nouvel environnement. Une migration implique la plupart du temps des volumes de données très importants et celles-ci sont souvent très hétérogènes. De plus, il faut garder une certaine cohérence entre le nouveau et l'ancien système. Les outils ETL sont donc bien adaptés à cette utilisation.

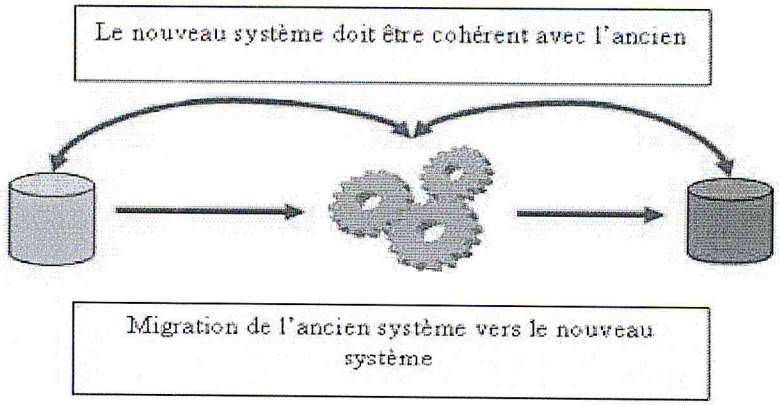


Figure 18: Schéma explicatif de la migration de données [13].

4.2 Synchronisation de données

Dans de nombreux systèmes, les données sont gérées séparément par de multiples applications. La synchronisation de celles-ci permet de maintenir une cohésion entre toutes les applications et les bases de données.

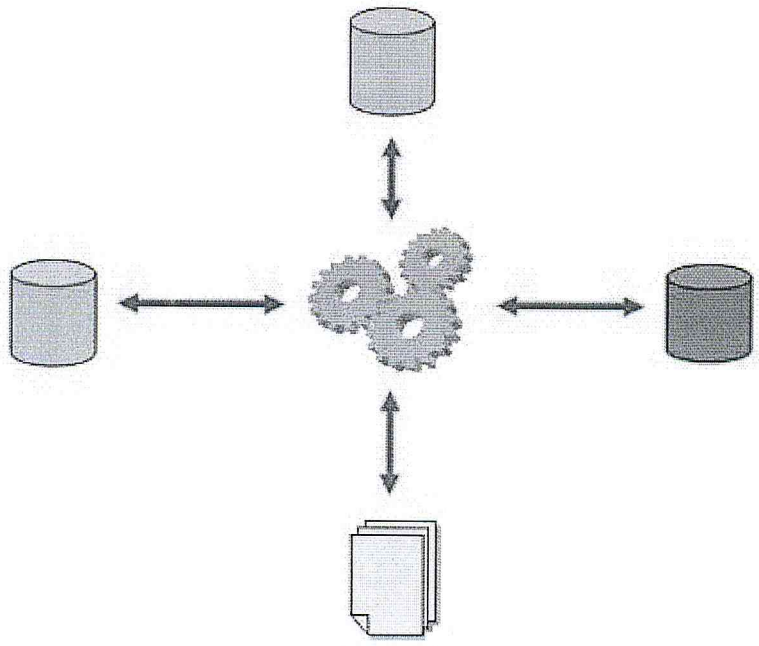


Figure 19 : Schéma explicatif de la synchronisation de données [13].

5 Evolution d'ETL avec la croissance de données « Big Data »

À la fin des années 1980, les premières piles de données de système décisionnel ont commencé à se matérialiser, et ils ressemblent généralement à la figure 15.

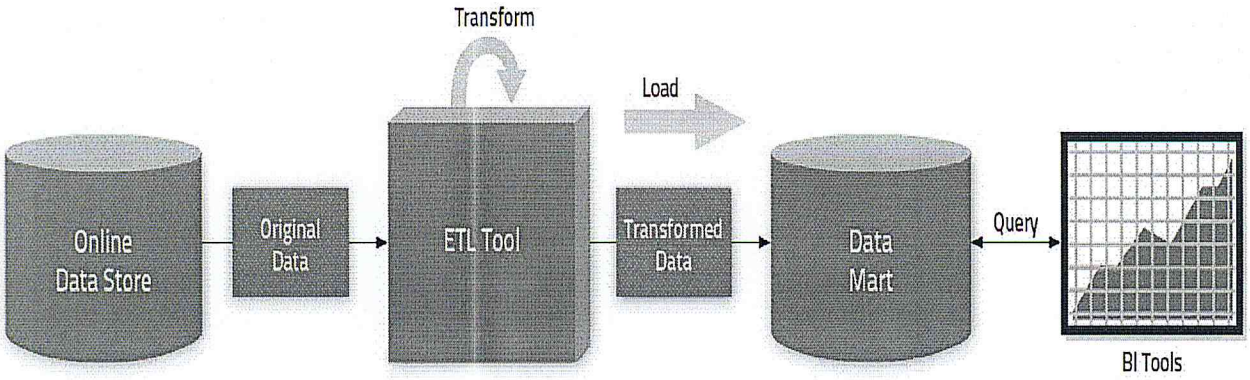


Figure 20 : Architecture décisionnelle Traditionnelle [W02].

Dans cette configuration, l'outil ETL est chargé d'extraire les données des systèmes sources (Extraction), puis il fait la transformation sur ces données (Transformation), puis il charge les données dans le datamart cible (Chargement). Outil ETL traditionnel à quatre composantes principales:

1. Connecteurs de lecture et écriture de ou vers de nombreux types de sources et cibles.
2. Le logique métier et les métadonnées qui déterminent de quelles transformations doivent être réalisés.
3. Bibliothèques de transformateurs disposant d'une panoplie de fonctions de transformations
4. Un moteur chargé d'exécuter les différentes transformations prévues dans la phase T.

A la fin des années 1990, la taille des données a connu une évolution au-delà de la capacité des outils ETL, ce qui a engendré une nette dégradation dans les performances des outils ETL. Les éditeurs ont basculé de l'ETL vers l'ELT (Voir figure 21).

Les choses se sont encore compliquées lorsqu'il s'agissait de traiter des cas d'erreurs. Par exemple, suite à des erreurs découvertes dans la logique du processus ETL pour les trois derniers mois, toutes les transformations effectuées durant cette période doivent être ré-exécutées avec les correctifs nécessaires. Malheureusement, Ce qui prend des semaines à

corriger sachant qu'en parallèle d'autres quantités de données arrivent dans la file du processus ETL. En résumé, les moteurs de transformation de type ETL sont devenus un goulot d'étranglement important dans la chaîne décisionnelle.

Le processus ELT est semblable à l'ETL mais avec une différence principale: les phases d'extraction et de chargement sont isolées de la phase de transformation. Cela a un certain nombre d'avantages.

Les données sont extraites des sources de données et chargées d'abord dans une base intermédiaire "*Staging Database*". Les contrôles des règles d'intégrité et les corrections pertinentes peuvent être appliqués dans la zone de transfert. Les données sources sont ensuite chargées dans l'entrepôt. A ce niveau, les transformations sont exécutées pour refaçonner les données dans le format cible destiné pour l'analyse en ligne.

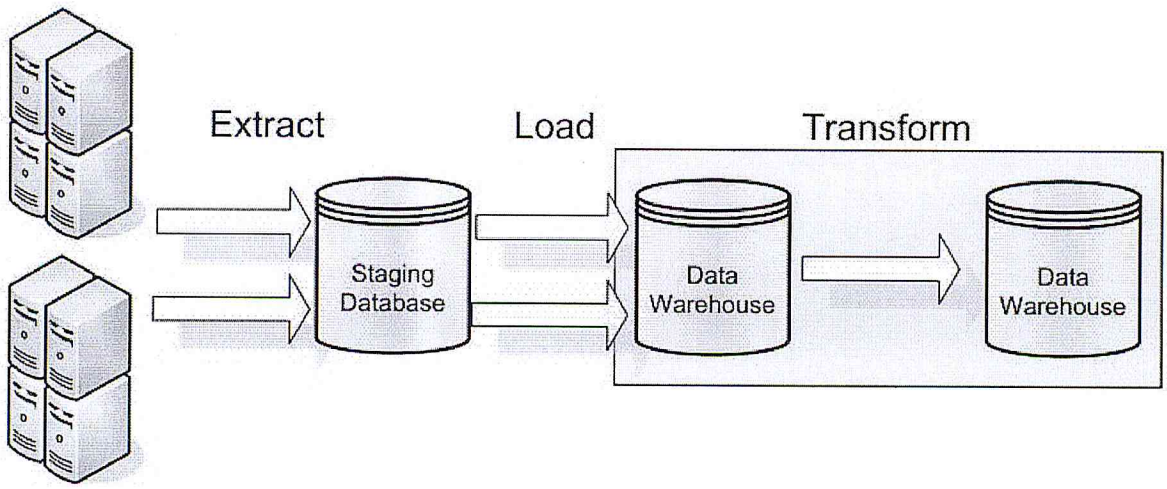


Figure 21 : Architecture de ELT [14]

a. Avantages d'ELT

a.1 La flexibilité

En général, dans la mise en œuvre d'un ELT, les phases d'extraction et de chargement sont isolées de la phase de transformation. Cela signifie que les futurs besoins peuvent être facilement incorporés dans la structure d'entreposage.

a.2 Minimisation des risques

Retrait des interdépendances entre chaque étape du processus de ELT permet au processus de développement d'être isolé, et la conception de processus individuel peut

donc aussi être isolé. Ceci fournit une excellente plate-forme pour le changement et la maintenance.

b. Inconvénients d'ELT

- ✓ **Contre la norme :** ELT est une approche émergente pour l'entreposage. Bien qu'il a fait preuve par son utilisation abondante dans les implémentations à travers le monde, l'ELT nécessite un changement de mentalité et d'approche de conception contre les méthodes traditionnelles.
- ✓ **Disponibilité des outils :** ELT souffre d'une disponibilité limitée d'outils.
- ✓ **Perdre les statistiques :** Avec l'approche ELT on perd les statistiques d'exécution et le suivi de lignage des données, en particulier les analyses d'impact de métadonnées sur les modifications apportées au fichier disparate, une table ou sources non structurées.

Le passage à l'ELT a fait ses preuves pendant des années, puis plusieurs systèmes de RDBM ont commencé à évoluer vers des architectures MPP (Massively Parallel Processing) qui s'exécute à travers un certain nombre de serveurs. Le parallélisme fait que le SGBDR est un bon endroit pour faire la transformation par rapport à l'outil ETL. A noter que certains fournisseurs d'outils ETL ont essayés de paralléliser leurs systèmes, mais peu réussi.

Cependant, les SGBDR ont été créés pour faire des requêtes, et non pour les transformations par lots (mode batch). La volumétrie des données qui n'a cessé d'accroitre a fait perdre les performances des requêtes dans les SGBDR.

Suite à cette situation problématique, une nouvelle solution a vu le jour, il s'agit du framework Apache Hadoop destiné pour un stockage de données distribuées et un traitement parallèle à grande échelle en utilisant du matériel standard de l'industrie. En outre, Hadoop a été conçu pour être très flexible et traiter n'importe quel type de données, indépendamment de la structure.

Maintenant que Hadoop est disponible, des centaines d'organisations se déplacent vers l'architecture massivement parallèle qui repose sur la distribution du stockage et des traitements sur une *grille* de serveurs (cluster). Les données sont stockées par blocs et répliquées entre les serveurs et les traitements (script SQL et code de calculs) sont transférés sur les serveurs impliqués dans le traitement. La donnée ne bouge pas d'un serveur à l'autre,

c'est le code de traitement (d'un volume toujours très faible) qui se déplace. Ce principe s'appelle la colocalisation entre traitements et données.

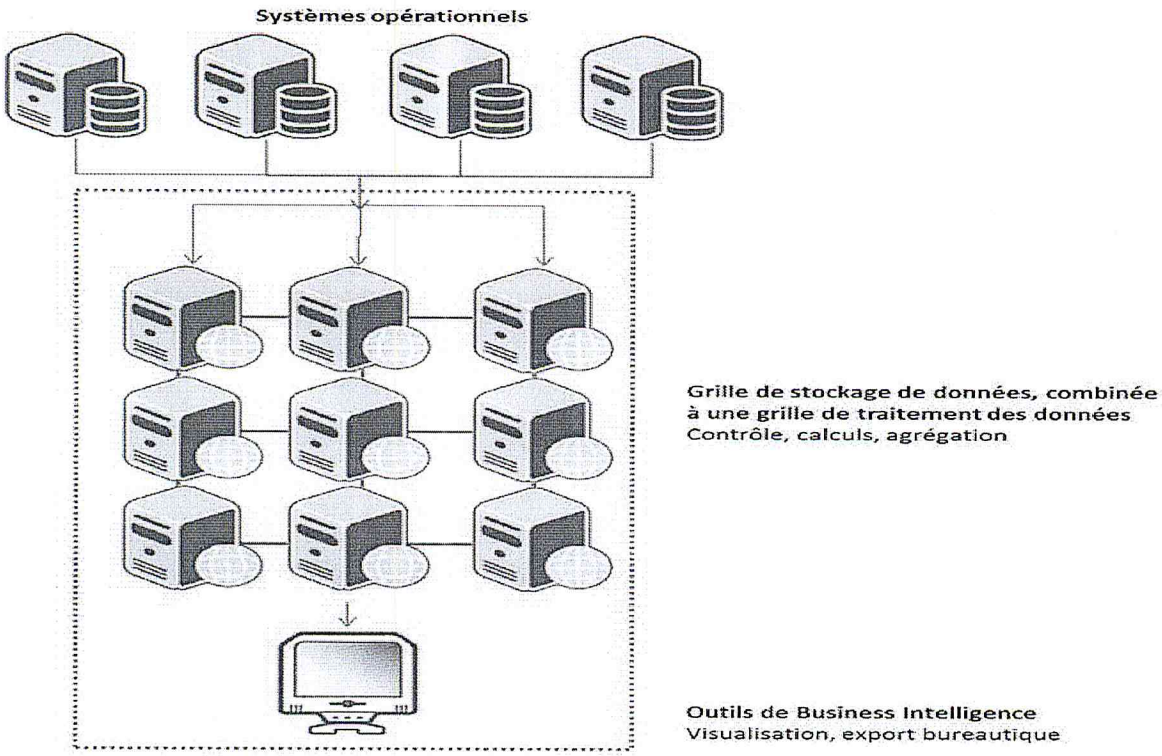


Figure 22 : Architecture Massivement Parallèle [W03].

Cette architecture permet de stocker une quantité de données immense (sans limites) et de manière élastique. Plus la taille de la grille augmente, plus sa capacité de traitement augmente, elle est aussi très tolérante aux pannes. Par exemple, si un processus nécessite neuf heures d'exécution sur 20 serveurs, et qu'à la huitième heure quatre de ces serveurs tombent en panne, l'exécution du processus continue normalement, pas besoin de refaire à partir de zéro. Si suite à une erreur décelée dans la logique ETL, il est nécessaire de relancer le processus pour les trois derniers mois, il est possible dans une telle architecture de rajouter temporairement quelques nœuds au cluster pour obtenir une vitesse de traitement supplémentaire.

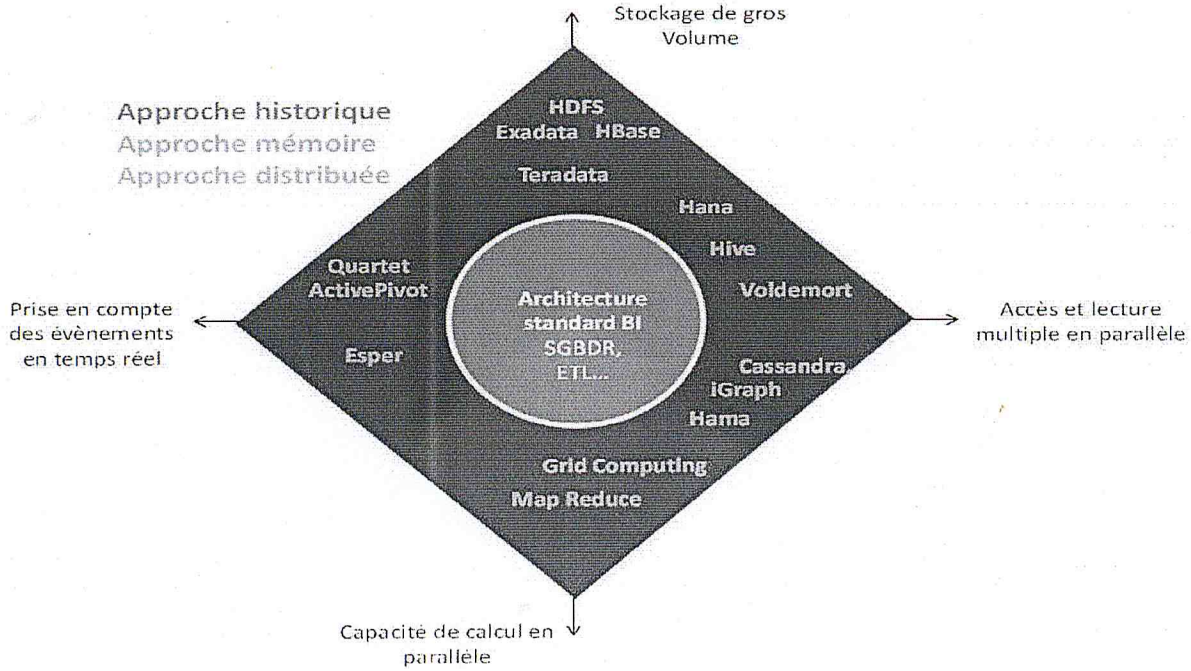


Figure 23 : solutions par classe de contrainte de performance [W03].

6 Conclusion

Dans ce chapitre nous avons vu les fonctionnalités et les différentes phases de l'ETL ainsi leur domaine d'applications, nous constatons un défis majeur pour les outils ETL depuis leur apparition c'est la croissance des données, Afin qu'il puisse répondre à ces défis en terme de performance, il a commencé à évoluer vers des architectures parallèles et des modèle de traitement parallèle comme le MapReduce ce qui donne une grande évolution dans les systèmes décisionnels. Dans le chapitre suivant nous allons étudier le phénomène de Big Data et le paradigme MapReduce.

Partie 2: Big data ET MapReduce

Chapitre III: Big Data

Chapitre IV: MapReduce

Chapitre V: Processus ETL et modèle MapReduce

Chapitre III

Big Data

Chapitre III : Big Data

La quantité de données dans notre monde a explosé, et l'analyse de grands ensembles de données – les Big Data – va devenir une base clé de la concurrence, qui sous-tend de nouvelles vagues de croissance de la productivité, l'innovation et le surplus du consommateur. Les dirigeants de tous les secteurs devront faire face aux conséquences de gros volumes de données, et pas seulement quelques gestionnaires orientés données. Dans ce chapitre, nous allons parler sur le phénomène des big data, leurs origines, caractéristiques et défis.

1. Définition

Big data (ou les grandes données) se réfère à des ensembles de données dont la taille est au-delà de la capacité des outils logiciels de base de données typiques pour capturer, stocker, gérer et analyser. [16]

2. L'origine des données des Big Data

Les données traitées par le Big Data proviennent notamment :

• du Web: journaux d'accès, réseaux sociaux, e-commerce, indexation, stockage de documents, de photos, de vidéos, linked data, etc. (ex: Google traitait 24 petaoctets de données par jour avec MapReduce en 2009).

• plus généralement, de l'internet et des objets communicants: RFID, réseaux de capteurs, journaux des appels en téléphonie;

• des sciences: génomique, astronomie, physique subatomique (ex: le CERN annonce produire 15 petaoctets de données par an avec le LHC), climatologie (ex: le centre de recherche allemand sur le climat gère une base de données de 60 petaoctets), etc.;

• données commerciales (ex: historique des transactions dans une chaîne d'hypermarchés);

• données personnelles (ex: dossiers médicaux);

• données publiques (open data). [17]

3. Caractéristiques des Big data

Voici les quatre caractéristiques clés qui définissent les grandes données :

a. Volume : les données générées par la machine sont produites en quantités beaucoup plus grandes que les données non traditionnelles. Par exemple, un moteur de jet peut générer 10 To de données en 30 minutes. Avec plus de 25.000 vols des compagnies aériennes par jour, le

volume quotidien de cette source de données seulement se heurte à des pétaoctets. Les compteurs intelligents et les équipements industriels lourds tels que les raffineries de pétrole et appareils de forage génèrent des volumes de données similaires, ce qui aggrave le problème.

b. Vélocité : Les flux de données des médias sociaux - tout en n'étant pas aussi massive que les données générées par une machine - produisent un grand afflux d'opinions et de relations précieuses à la gestion de la relation clientèle. Même à 140 caractères par tweet, la haute vitesse (ou fréquence) des données de Twitter assure des volumes importants (plus de 8 To par jour).

c. Variété : les formats de données traditionnelles ont tendance à être relativement bien décrits et évoluent lentement. En revanche, les formats de données non traditionnelles présentent un taux vertigineux des changements. Quand de nouveaux services sont ajoutés, de nouveaux capteurs sont déployés, de nouvelles campagnes marketing sont exécutées ; de nouveaux types de données sont nécessaires pour capturer les informations résultantes.

d. Valeur : La valeur des différentes données varie considérablement. Généralement, il y a une bonne information cachée parmi un plus grand nombre de données non traditionnelles, le défi consiste à identifier ce qui est précieux et ensuite transformer et extraire les données pour l'analyse. [18]

4. Quelques technologies des Big Data

Il y a un nombre croissant de technologies utilisées pour agréger, manipuler, gérer, et analyser les données volumineuses. Voici quelques-unes de ces technologies les plus éminentes :

a. Intelligence d'affaires (BI : Business intelligence) : Un type de logiciel d'application conçu pour rapporter, analyser et présenter des données : Les outils de BI sont souvent utilisés pour lire les données qui ont déjà été stockées dans un data mart d'un entrepôt de données. Les outils de BI peuvent également être utilisés pour créer des rapports standards qui sont générés de façon périodique, ou pour afficher des informations sur des tableaux de bord de gestion en temps réel, c.-à-d. des affichages intégrées des critères qui mesurent la performance d'un système.

b. Big Table : Grande table ; est un système de base de données distribuées construit sur le Google File System. C'est une inspiration pour HBase.

c. L'entrepôt de données : est une base de données spécialisée optimisée pour les rapports, souvent utilisée pour stocker des grandes quantités de données structurées. Les données sont chargées en utilisant les outils d'ETL (Extraction, Transformation et Load) à partir des magasins de données opérationnels. Les rapports sont souvent générés en utilisant des outils d'intelligence d'affaires.

d. Extraire, transformer et charger (ETL) : des outils logiciels utilisés pour extraire les données provenant des sources extérieures, les transformer pour répondre aux besoins opérationnels, et les charger dans une base de données ou un entrepôt de données:

e. SQL: A l' origine, c'est un acronyme pour le langage structuré des requêtes, SQL est un langage informatique conçu pour la gestion des données dans les bases de données *relationnelles*. Cette technique offre la possibilité d'insérer, d'interroger, de mettre à jour et de supprimer les données, ainsi que de gérer les schémas de données (structures de base de données) et de contrôler l'accès aux données dans la base de données. [19]

5. Défis de stockage des Big data :

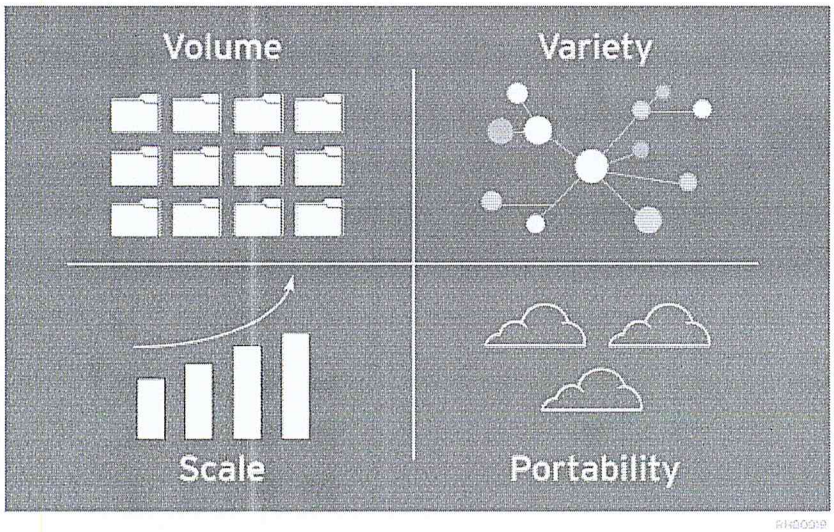


Figure 24 : Les défis du stockage des Big data

a. Volume : Le volume de données à l'intérieur de l'entreprise se développe à un rythme alarmant et les entreprises sont de plus en plus sous les pressions légales et réglementaires pour conserver leurs données plus longtemps. Pour de nombreuses entreprises, en particulier celles de l'industrie qui font face aux consommateurs, la valeur commerciale potentielle énorme des vastes quantités de données clientes exige que de telles données ne devraient jamais être jetées.

Les approches traditionnelles de stockage, tel que le stockage attaché au réseau (Network-

Attached Storage : NAS) et le réseau de zone de stockage (Storage Area Network : SAN), s'appuient sur des composants propriétaires coûteux n'ont pas été conçues avec les volumes de données d'aujourd'hui. Tenter de suivre la croissance des données en utilisant les systèmes de stockage traditionnels est inefficace et coûteux, et se traduit par une infrastructure de stockage inflexible qui est difficile à gérer à grande échelle.

b. Variété : Dans le passé, les entreprises étaient surtout préoccupées par l'analyse de volumes de données transactionnelles relativement uniformes afin de prendre des décisions commerciales plus efficaces. Aujourd'hui, les données viennent dans une grande variété de formats (par exemple, e-mail, documents, journaux d'application, données des compteurs ou des capteurs, données d'objets, images, vidéo, etc.) qui peuvent offrir des indications précieuses pour l'entreprise.
La variété des données étant actuellement générée, et la nécessité de stocker et analyser efficacement de gros volumes obligent de nombreuses entreprises à envisager de nouvelles approches pour le stockage.

c. Evolutivité : Il y a dix ans, le responsable informatique pouvait simplement estimer la quantité maximale de stockage dont il avait besoin et acheter de l'espace disque en conséquence. Aujourd'hui, avec des entreprises générant 40-60%, ou plus, des données supplémentaires chaque année, prévoir une taille maximal de stockage n'est plus possible, et acheter des quantités massives de capacité de stockage à l'avance serait du gaspillage. A Monolithic, des systèmes de stockage scale-up (d'évolutivité), l'évolutivité est limitée à partir du moment de l'achat, et la planification rigoureuse des capacités à l'avance est donc nécessaire.

d. Portabilité : Bien que les silos de données isolés ne sont pas un nouveau défi pour l'informatique, big data a rendu les solutions aux problèmes classiques obsolètes, en raison de la largeur de la bande des réseaux étendus (WAN) réduite, et du volume considérable des données concernées.
En outre, en raison de la popularité croissante des nuages informatiques (cloud computing) publics et privés comme alternatives aux centres de données (datacenters) traditionnelles, les solutions de stockage de données d'aujourd'hui doivent être portables sur tous les environnements de déploiement possibles. [19]

6. Statistiques :

Jetons un œil sur les statistiques du Big data en monde [W04] :

- 90% des données mondiales sont créées les 2 dernières années.
- Quotidiennement, les utilisations créent 2.5 quintillion octet de données.
- Facebook : 500 téraoctets traités chaque jour.
- Google : 50 billions de pages indexées chaque jour.
- YouTube : 60 heures de vidéos chargées chaque minute.
- Twitter : 340 millions de tweets postés chaque jour.

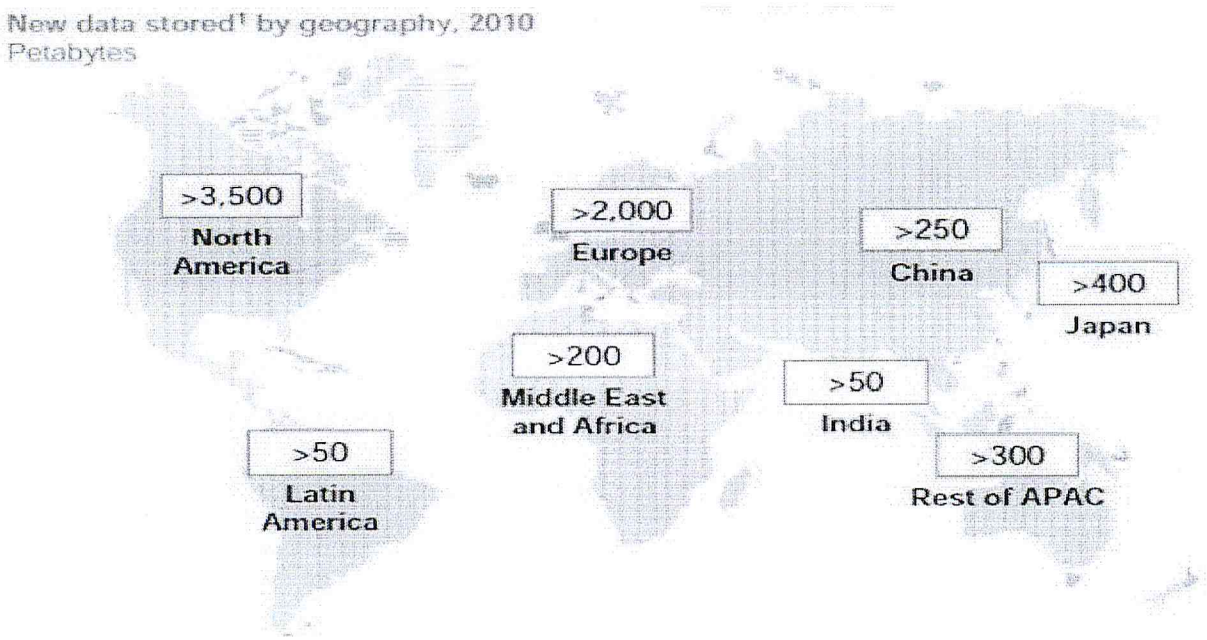


Figure 25 : Quantités de données stockées géographiquement [20].

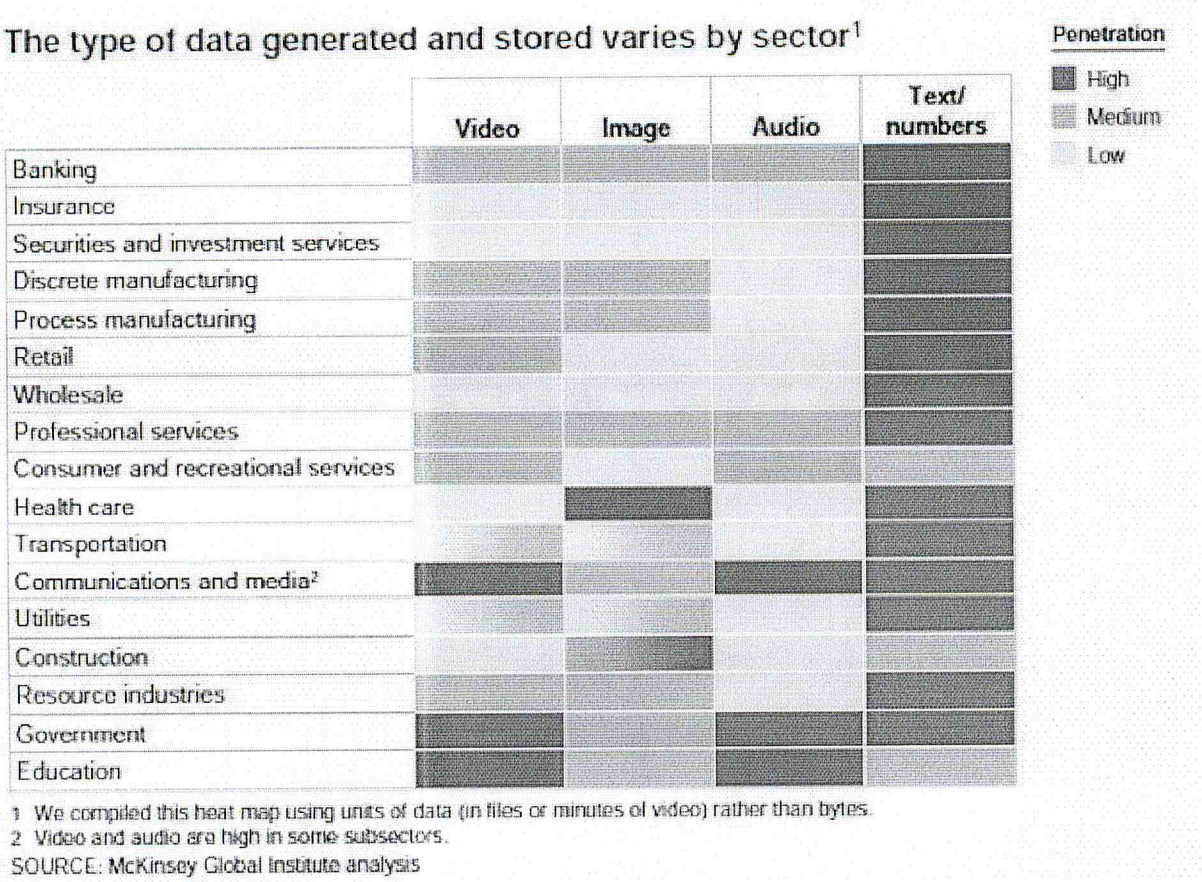


Figure 26 : Les stockages de données par secteur [20]

7. Mesure

Comment est mesuré le Big data :

1 gigaoctet = 1000 megaoctet (10^9 megaoctet).

1 téraoctet = 1000 gigaoctet (10^{12} megaoctet).

1 pétaoctet = 1000 téraoctet (10^{15} megaoctet).

1 exaoctet = 1000 pétaoctet (10^{18} megaoctet).

1 zetta octet = 1000 exaoctet (10^{21} megaoctet).

8. Conclusion

L'augmentation du volume et du détail des informations saisies par les entreprises, la montée du multimédia, les médias sociaux et l'Internet sont des choses qui vont alimenter la croissance exponentielle des données pour l'avenir prévisible. Le traitement de telles données nécessitera alors de nouvelles capacités logicielles et matérielles plus efficaces et intelligentes. Nous allons introduire dans le chapitre suivant un modèle de calcul parallèle ayant connu une vaste utilisation par des entreprises mondiales de haut niveau telles que : Google, Microsoft et Facebook : c'est le MapReduce.

Chapitre IV

MapReduce

Chapitre IV: MapReduce

1. Définition

MapReduce est un modèle de programmation pour exprimer des calculs distribués sur des quantités massives de données ; et un framework d'exécution pour le traitement des données à grande échelle sur des clusters de serveurs des produits de base (commodity). Il a été initialement développé par Google (2004) et fondé sur des principes bien connus dans le traitement parallèle et distribué. [21]

Plus brièvement, MapReduce est un modèle de programmation pour le traitement des grandes masses de données avec un algorithme parallèle et distribué dans un cluster.

Ce modèle est en train de connaitre un vif succès auprès des sociétés possédant d'importants datacenters telles qu'Amazon ou Facebook, il commence aussi à être utilisé au sein du Cloud computing. Google l'a utilisé pour régénérer tout son index du Word Wild Web.

2. Principe

Il s'agit de décomposer une tâche en tâches plus petites, ou plus précisément découper une tâche portant sur de très gros volumes de données en tâches identiques portant sur des sous-ensembles de ces données. Les tâches (et leurs données) sont ensuite dispatchées vers différents serveurs, puis les résultats sont récupérés et consolidés. La phase de décomposition des tâches est la partie Map, tandis que la phase de la consolidation des résultats est la partie Reduce. [22] Chaque phase manipule des paires (clé,valeur) en entrée et sortie.

Un cluster MapReduce utilise une architecture de type Maître-esclave où un nœud maître dirige tous les nœuds esclaves.

3. Map

Dans la phase Map, le nœud maitre prend les donnée d'entrée, il les divise à des sous-ensembles de données sous la forme des listes des paires (clé, valeur) et les distribue sur les nœuds «travailleurs» appelés Mappers. Chaque nœud Mapper accepte en entrée le sous-ensemble de donnée qui lui avait été attribué. Il les traite selon un contexte donné. Le résultat sera une liste de paires (clé, valeur) ayant subisses des différents traitements.

$$Map(clé1,valeur1) \rightarrow list(clé2,valeur2)$$

4. Reduce

Vient ensuite la phase Reduce. Les Mappers font remonter leurs résultats au nœud parent qui les avait sollicités. Celui-ci trie ces résultats (ensemble de paires : clé, valeur) et les agrège de façon qu'il associe toutes les valeurs correspondantes à la **même** clé à une **unique** paire (clé, valeur) :

(clé1,val1), (clé2,val2) , (clé1,val3), (clé2,val4) → (clé1, list(val1,val3)), (clé2, list(val2,val4)

Cette opération est appelée *Shuffle*. Ces paires, constituant le résultat intermédiaires, vont être reçues par des nœuds spéciaux appelés Reducers. Un Reducer prend en entrée les paires (clé, liste de valeurs) puis, il itère sur les valeurs de la liste. Il fait souvent les calcules sur les agrégations : le nombre des valeurs associées, l'addition de ces valeurs, leur max/min, etc.

Le maitre mémorise l'état (inactif, en cours ou terminé), et l'identité des nœuds.

À la fin du processus, le nœud d'origine peut recomposer une réponse au problème qui lui avait été soumis :

Reduce(key,list(val)) → list(valeur)

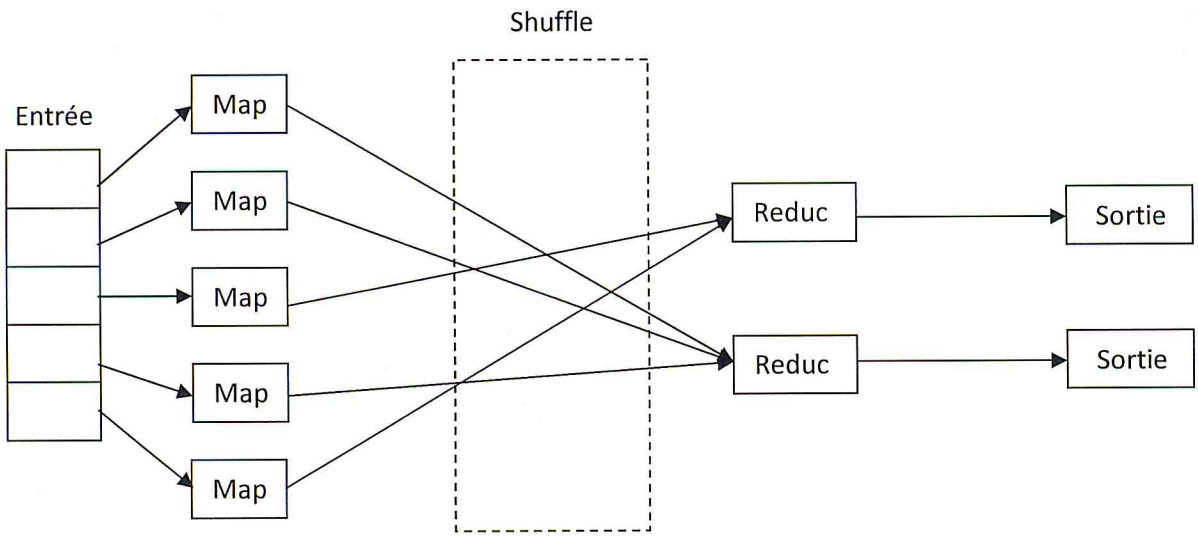


Figure 27 : Aperçu général sur les étapes de MapReduce

a. Exemple 1 : Website visits

Pour visualiser la façon dont le MapReduce fonctionne, en voici un exemple :

On veut savoir d'où proviennent les visiteurs d'un site web donné. On a un fichier source contenant le nom du pays (défini par l'adresse IP par exemple) et le temps de la visite :

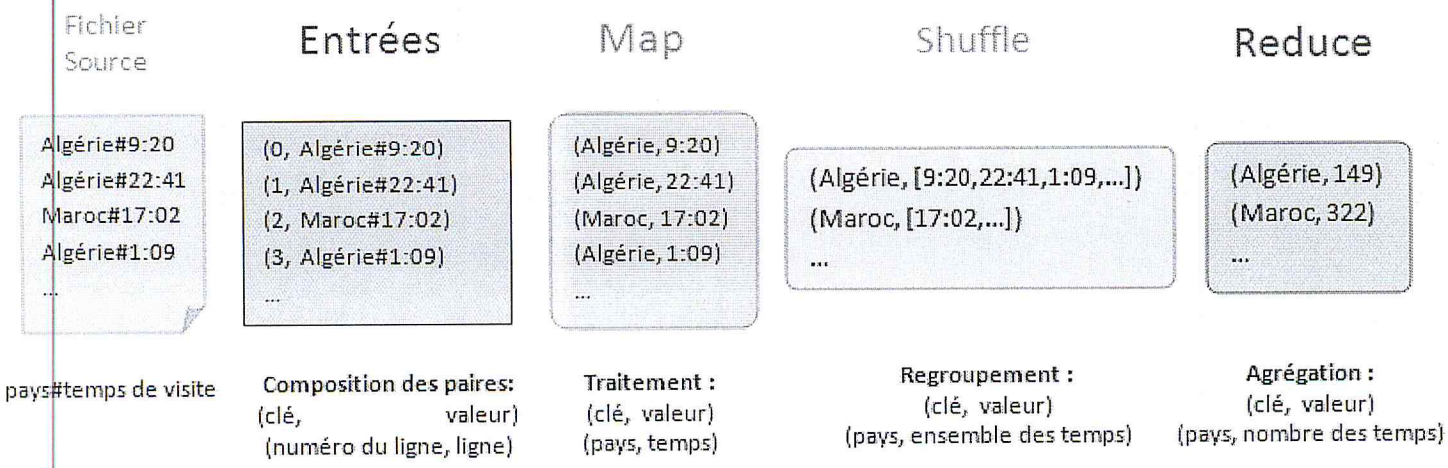


Figure 28 : Exemple d'un processus ETL Website visits

b. Exemple 2 : Word Count

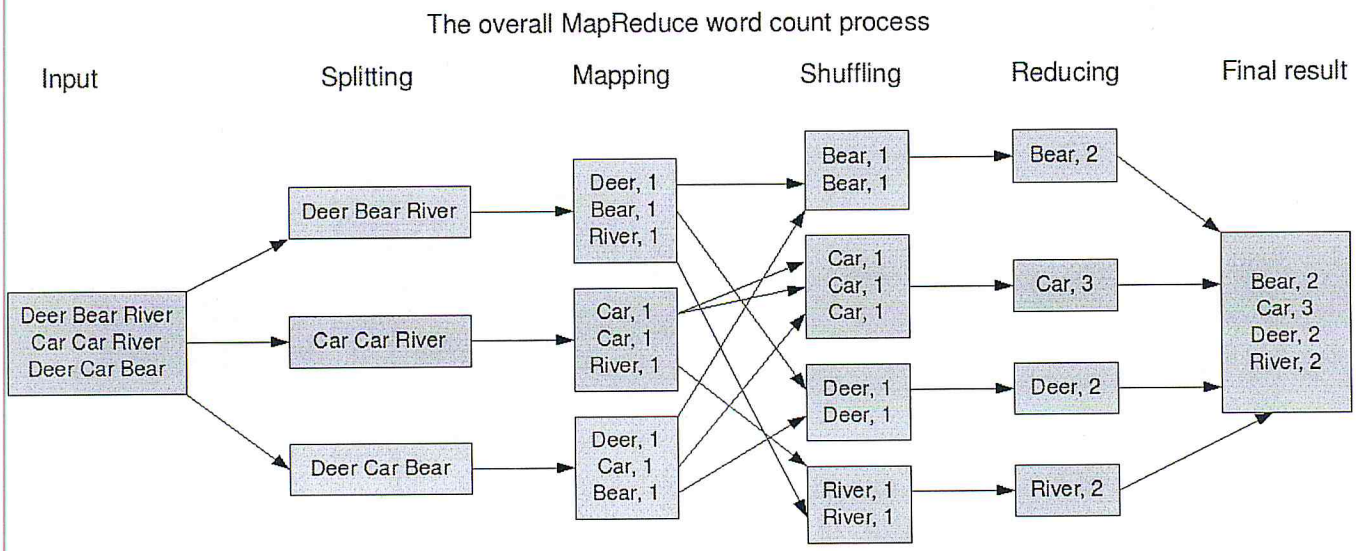


Figure 29 : Exemple d'un processus ETL Words Count [W05]

La normalisation autour de ces principes s'appuie sur la manipulation de couples (clé,valeur). La tâche initiale est un couple (clé,valeur), et chacune des tâches intermédiaires aussi. Chaque tâche intermédiaire retourne un résultat sous la forme d'un couple (clé,valeur), et la fonction Reduce combine tous ces résultats en un couple (clé,valeur) unique.

Chapitre IV: MapReduce

L'algorithme MapReduce est principalement destiné à des tâches de type batch, de très grande ampleur (large-scale), portant sur de très grands volumes de données. La décomposition typique consiste à découper le volume de données initial en N volumes plus petits, qui peuvent être traités séparément par la fonction Map. L'exemple souvent pris pour illustration est le comptage du nombre d'occurrence de chaque mot d'un très grand fichier. Le grand fichier est décomposé en plus petits fichiers, et les occurrences sont comptées sur chacun d'eux. La fonction Reduce additionne les décomptes obtenus sur chaque fichier intermédiaire.

5. Enchainement des tâches MapReduce

Tous les problème ne peuvent être résolus avec un seul programme MapReduce, mais moins encore sont ceux qui peuvent être résolus avec un seul programme MapReduce. Beaucoup de problèmes nécessitent plusieurs étapes MapReduce s'exécutant en série pour atteindre l'objectif :

Map1 -> Reduce1 -> Map2 -> Reduce2 -> Map3 ...

6. Caractéristiques

MapReduce possède quelques caractéristiques [23] :

- Le modèle de programmation du MapReduce est simple mais très expressif. Bien qu'il ne possède que deux fonctions, map() et Reduce(), elles peuvent être utilisées pour de nombreux types de traitement des données, les fouilles de données, les graphes… Il est indépendant du système de stockage et peut manipuler de nombreux types de variable.
- Le système découpe automatiquement les données en entrée en bloc de données de même taille. Puis, il planifie l'exécution des tâches sur les nœuds disponibles.
- Il fournit une tolérance aux fautes à grain fin grâce à laquelle il peut redémarrer les nœuds ayant rencontré une erreur ou assigner la tâche à un autre nœud de manière transparente.
- La parallélisation est invisible à l'utilisateur afin de lui permettre de se concentrer sur le traitement des données [24]

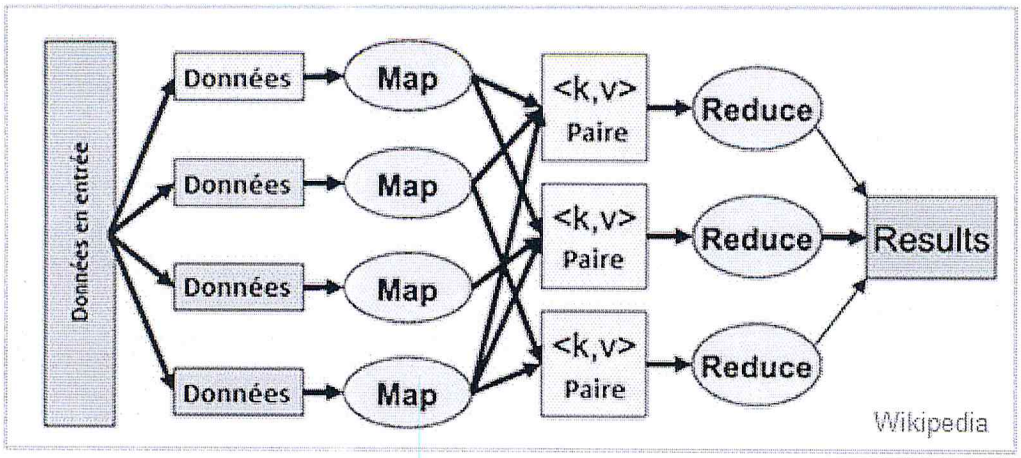


Figure 30 : Schéma de fonctionnement du MapReduce

Une fois qu'un nœud a terminé une tâche, on lui assigne un nouveau bloc de données. Grâce à cela, un nœud rapide fera beaucoup plus de calculs qu'un nœud plus lent. Le nombre de tâches Map ne dépend pas du nombre de nœuds, mais du nombre de blocs de données en entrée. Chaque bloc se fait assigner une seule tâche Map. De plus, toutes les tâches Map n'ont pas besoin d'être exécutées en même temps en parallèle, les tâches Reduce suivent la même logique. Par exemple, si des données en entrée sont divisées en 400 blocs et qu'il y a 40 nœuds dans le cluster, le nombre de tâches Map sera de 400. Il faudra alors 10 vagues de Map pour réaliser le Mapping des données [24] [25].

7. Points faibles

Le MapReduce est apparu en 2004, c'est une technologie encore jeune qui possède quelques points faibles [24] :

- Il ne supporte pas les langages haut-niveau comme le SQL
- Il ne gère pas les index. Une tâche MapReduce peut travailler après que les données en entrée soient stockées dans sa mémoire. Cependant, MapReduce a besoin de parser chaque donnée en entrée afin de la transformer en objet pour la traiter, ce qui provoque des baisses de performance.
- Il utilise un seul flot de données. Le MapReduce est facile à utiliser avec une seule abstraction mais avec un flot de donnée fixe. Par conséquence, certains algorithmes complexes sont difficiles à implémenter avec seulement les méthodes map() et Reduce(). De plus, les algorithmes qui requièrent de multiples éléments en entrée ne

sont pas bien supportés car le flot de données du MapReduce est prévu pour lire un seul élément en entrée et génère une seule donnée en sortie.

- Quelques points peuvent réduire les performances du MapReduce. Avec sa tolérance aux pannes et sa scalabilité, les opérations du MapReduce ne sont pas toujours optimisées pour les entrées/sorties. De plus, les méthodes map() et Reduce() sont bloquantes. Cela signifie que pour passer à l'étape suivante, il faut attendre que toutes les tâches de l'étape courante soient terminées. Le MapReduce n'a pas de plan spécifique d'exécution et n'a pas optimisé le transfert de données entre ces nœuds.[24]

8. Conclusion

A partir de ce que l'on a vu, le MapReduce est une abstraction très utile dans l'air du big data sur lequel le profit du parallélisme et la programmation fonctionnel deviendrais un atout.

Bien que MapReduce soit basé principalement sur les deux fonctions : Map et Reduce, il repose bel et bien sur d'autres opérations qui sont aussi nécessaire pour son fonctionnement, notamment la lecture des entrés, la répartition des sous-ensembles des donné sur les Mappers, l'écriture sur le disque local etc. On détaillera ces opérations dans les chapitres suivants.

Le principe est simple donc, mais l'apport de l'algorithme MapReduce est de bien conceptualiser, en vue de normaliser les opérations d'intendance de sorte qu'un même framework puisse prendre en charge une diversité de tâches partitionnables, ce qui nous permettra de nous concentrer sur les traitements proprement dits, tandis que le framework prend en charge la logistique de la répartition.

Chapitre V

Processus ETL et modèle MapReduce

Chapitre V : Processus ETL et modèle MapReduce

Comme nous l'avons détaillé dans le 2er chapitre, l'ETL fait la collecte des données à partir de sources diverses, applique des différentes opérations de transformations définies par l'utilisateur sur ces données puis les consolide et stocke. Les ETL traditionnels font face à de nombreux défis lorsque les données deviennent énormes, des centaines de Giga-octets (si l'on considère un environnement d'une activité moyenne) à traiter et analyser chaque jour. Le problème majeur c'est le temps. La volumétrie des données alourdit énormément le processus ETL pendant que la demande des utilisateurs ne cesse d'augmenter. L'utilisation du parallélisme est une solution adéquate pour une meilleure performance en termes de temps et d'évolutivité (scalabality). Ce qui a donné naissance à de nombreuses solutions, entre autres, le fameux MapReduce qui a été largement utilisé pour le calcul parallèle dans des environnements de big data.

MapReduce est basé principalement sur les deux opérations : Map et Reduce. Il y a plusieurs Maps qui s'exécutent en parallèle, elles reçoivent des données extraites à partir des fichiers sources pour appliquer des traitements spécifiques. Les données sorties des Mappers sont envoyées ensuite au Reducers si des agrégations sont nécessaires. Les données qui ont subi des traitements sont finalement consolidées et stockées. Nous remarquons que cette logique s'adapte parfaitement avec le processus ETL qui suit exactement la même chaine : l'extraction des données, puis une série de transformations sur ces données : nettoyages, agrégations, jointures, etc. puis consolidation et stockage des données dans un cube ou un entrepôt de données, avec le plaisir du parallélisme.

L'objectif est alors d'appliquer la logique du MapReduce pendant les trois phases du processus ETL : l'extraction, la transformation puis le chargement.

1. Extraction

Le modèle MapReduce propose une opération initiale de partitionnement, avant l'étape Map, consiste à diviser le fichier source en plusieurs sous-ensembles de données, chacun d'eux étant ensuite affecté à un Mapper (Les sources sont supposées orientées-ligne, comme le cas des fichiers textes par exemples, les sources structurées doivent passer par une opération de conversion). Une approche appelée partitionnement **logique** s'adapte au parallélisme. Il s'agit de lire la taille du fichier en entrée, puis la diviser sur le nombre du Map à utiliser. Cette opération nous permet de définir une liste d'identificateurs des partitions logiques sous forme de triplets (f,d,l) qui servent comme références des partitions dans le fichier source, où **f** est le fichier source, **d** est le début de la partition dans le fichier source (appelé souvent start offset)

et **l** renseigne sur la longueur de la partition en nombre de lignes. Les Mappers quand ils se mettent au travail, ils utilisent les informations sur les partitions qui lui ont été attribuées ; ils prennent le fichier à lire, le début de la partition dans ce fichier et le nombre de lignes à parcourir.

2. Transformations

Les transformations, qui sont les contraintes de validité à respecter spécifiées par l'utilisateur, sont à effectuer soit à la phase Map ou Reduce. Les données sont présentées au Mapper ligne par ligne sous forme d'une paire (clé,valeur) où la clé étant la position de la ligne dans la source et la valeur étant la ligne lue (on peut y trouver plusieurs lignes séparées par un délimiteur connu au préalable) à partir du fichier source dans la limite de la partition logique associée. Le Mapper prend la ligne et applique sur elle une chaine de transformations puis la passe au Reducer pour continuer le reste du job qui peut appliquer d'autres transformations si elles existent. Par exemple chercher une valeur nulle, si elle existe, la remplacer par un zéro, ou sauter la ligne totalement. Autre exemple est de modifier une valeur par une autre, « Algérie » par « Alg » ainsi de suite. La majorité des transformations se font lors de la tâche Map, tandis qu'il existe d'autres qui se font au Reduce tel que la jointure et l'agrégation. Nous parlerons des types de transformation avec plus de détails dans le chapitre de la conception de notre système.

3 Chargement

Pour rendre cette phase parallèle, on doit avoir plusieurs Reducers qui écrivent toutes leur sorties sur le système de fichier.

4. ETLMR

On ne peut envisager de parler sur les environnements ETL sous MapReduce sans faire référence au travail de Liu, Thomson et Pederson (2011) ont présenté un framework ETL parallèle basé sur le paradigme MapReduce appelé ETLMR. Ils ont profité de la logique MapReduce pour fournir une plateforme ETL évaluative, tolérable aux pannes et très légère. Ils font un grand effort qui mérite être notre base de ce travail.

Ils ont présenté un certain nombre de nouvelles méthodes qui sont utilisées pour traiter les dimensions d'un schéma en étoile, des dimensions à flocons, SCD et des dimensions de données intensives. En outre, ils ont introduit le système de dimension hors-ligne qui s'évolue

mieux que le schéma de dimension en-ligne (décrit ci-après) lors de la manipulation des charges de travail énormes. Les évaluations montrent que ETLMR réalise une très bonne évolutivité et se compare favorablement avec d'autres outils d'entreposage de données MapReduce.

4.1 Présentation d'ETLMR

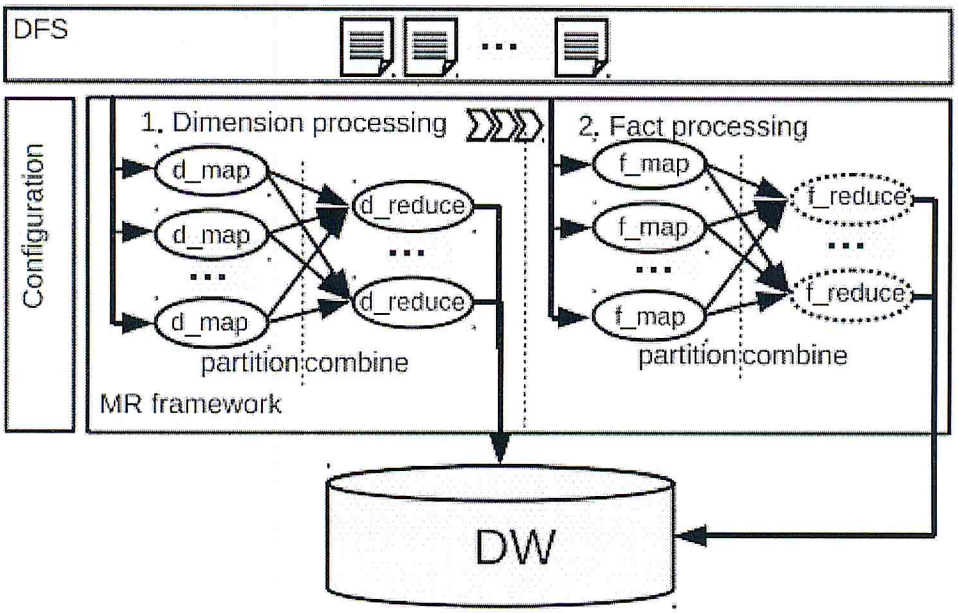


Figure 31 : Flux de données ETL dans un framework MapReduce [26]

La figure 33 illustre le flux de données en utilisant ETLMR sur MapReduce. Dans ETLMR, le traitement des dimensions se fait d'abord dans un job MapReduce, puis le traitement du fait est effectué dans un autre job MapReduce. Un job MapReduce engendre un certain nombre de tâches Map et Reduce parallèles pour traiter les données de dimension et de fait. Chaque tâche comporte plusieurs étapes, y compris la lecture des données à partir d'un système de fichiers distribués (DFS), l'exécution de la fonction de Map, le partitionnement, la combinaison de la sortie de Map, l'exécution de la fonction de Reduce et l'écriture des résultats dans le système de fichiers.

Dans le traitement des dimensions, les données d'entrée d'une table de dimension peuvent être traitées par différentes méthodes de traitement, par exemple, les données peuvent être traitées par une seule tâche, ou par toutes les tâches. Dans le traitement de fait, les données pour une table de faits sont partitionnées pour qu'elles soient traitées par un certain nombre de tâches parallèles. Cela comprend la recherche des clés de dimension et le

chargement des données de fait traitées dans le DW. Le traitement des données de fait dans les Reducers peut être omis (illustré par des ellipses en pointillés sur la figure 33) si aucune agrégation des données de fait n'est effectuée avant que ces données ne soient chargées.

Avant un job MapReduce commence, les ensembles de données provenant de systèmes de stockage hétérogènes sont divisés en plusieurs morceaux de tailles à peu près égales, qui sont distribués aux tâches Map/Reduce, puis traités dans des dimensions ou des faits. ETLMR emploie les deux méthodes de partitionnement suivantes :

1) Partitionnement Round Robin: Cette méthode distribue les lignes entre les tâches telle que la ligne numéro n est affectée à la tâche numéro (n modulo nb Map) où nb Map est le nombre des tâches Map. Il s'assure que les ensembles de données d'entrée sont équitablement répartis entre les tâches. Cette méthode est adaptée lorsque les données d'une dimension doivent être traitées par toutes les tâches.

2) Partitionnement hash de champ: Cette méthode désigne un ou plusieurs attributs comme des *attributs de partitionnement*. Les tuples (lignes) avec les mêmes valeurs de hachage sur les attributs de partitionnement sont affectés à une même tâche. S'il y a nb Map tâches, un tuple avec la valeur de hachage *hash* est affecté à la tâche numéro (*hash* modulo Maps nr). Cette méthode est adaptée lorsque toutes les lignes avec des valeurs identiques dans les attributs de hachage doivent être traitées par une même tâche.

ETLMR fournit des lecteurs de Maps en implémentant les deux méthodes de partitionnement ci-dessus. En outre, la plupart des frameworks MapReduce (Ex. Hadoop) offrent des différents types de lecteurs de sélections ou permettent aux utilisateurs de personnaliser leurs propres lecteurs.

Dans le traitement des dimensions, les données d'entrée sont traitées entre toutes les tâches Map/Reduce, utilisant des méthodes de traitement configurables. La plus fondamentale de ces méthodes consiste à configurer une dimension par tâche, par exemple, étant donné 3 dimensions et 3 tâches, chaque tâche traite les ensembles de données d'une seule dimension. Contrairement à cette méthode où seulement un nombre limité de tâches peut être utilisé, une autre approche est de laisser toutes les tâches de traiter une dimension donnée de telle sorte que chaque tâche traite des parties d'ensembles de données de la dimension. S'il existe des dépendances entre les dimensions, telles que les dimensions en flocons (ou normalisée), des méthodes de traitement spéciales (traitement level-wise ou hierarchy-wise) sont configurées

pour traiter l'ordre de traitement et la parallélisation des dimensions. Dans une autre optimisation, les dimensions peuvent être configurées pour être stockées de manière distribuée sur les nœuds, puis chargées dans le DW sur demande.

Dans le traitement des faits, chaque tâche Map/Reduce traite un ensemble de données de même taille, y compris la lecture des données, la recherche des valeurs clés des tables de dimension, la transformation et le chargement. Si un fait est un fait agrégé, les Reducers sont configurés pour le calcul des mesures sur toutes les lignes à l'aide des fonctions d'agrégation telles que *la somme*, *la moyenne* et *le comptage*. Pour optimiser les performances, Reducers peuvent être omises si aucune agrégation n'est nécessaire (ellipses en pointillés présentés dans la figure 33). De plus, les dimensions peuvent être lues entièrement ou partiellement dans la mémoire principale afin d'accélérer la recherche des clés de dimension, les *charges en vrac* (bulk-load)[1] sont utilisées pour transférer les données traitées à partir de la mémoire principale à la DW au moment de l'exécution.

L'algorithme suivant détaille tout le processus. Les opérations dans les lignes 2-4 et 6-7 sont les étapes MapReduce qui sont responsables de l'initialisation, en invoquant des jobs pour le traitement des dimensions et des faits, et pour le retour d'informations de traitement. Ligne 1 et 5 sont les étapes non-MapReduce qui sont utilisés pour la préparation des jeux de données d'entrée et pour la synchronisation de dimensions entre les nœuds (si aucun système de fichiers distribués DFS n'est installé).

L'algorithme 1 processus ETL sur un framework MapReduce :

1: partitionner les ensembles de données d'entrée;

2: lire les paramètres de configuration (tableau 1) et initialiser;

3: lire les données d'entrée et relayer les données de la fonction de Map dans le lecteur du Map;

4: Traiter les données de dimension et les charger dans des magasins de dimension en-ligne/hors-ligne;

5: Synchroniser les dimensions à travers les ordinateurs du cluster, le cas échéant;

6: Préparer le traitement de fait (se connecter et cacher les dimensions);

[1] Chargement en vrac est un moyen pour charger des données dans des «gros morceaux».

7: lire les données d'entrée pour le traitement des faits et effectuer des transformations dans les Mappers;

8: Charger en vrac les données dans le DW.

En ETLMR, tous les paramètres d'exécution sont stockés dans un seul fichier de configuration, y compris les paramètres de sources de données, les méthodes de partitionnement telles que les clés pour le partitionnement, les dimensions, les faits, les (grand) dimensions de données intensives, et le nombre de Mappers et Reducers. Le tableau suivant résume les paramètres importants de la configuration.

Ces paramètres donnent aux utilisateurs la flexibilité pour configurer les tâches pour être plus efficaces. Par exemple, si un utilisateur sait qu'une dimension est une dimension de données intensives, il peut l'ajouter à la liste *bigdims* de sorte qu'une méthode de traitement appropriée peut être choisie pour obtenir de meilleures performances et meilleures équilibrages de charge. [26]

Paramètres	Description
Dim_i	Définition de table de dimension, i = 1, . . ., n
$Fact_i$	Définition de table de faits, i = 1, . . ., m
Set_{bigdim}	Dimensions de données intensives dont les clés d'affaires sont utilisées pour le partitionnement des ensembles de données le cas échéant
Dim_i (a_0, a_1, ..., a_n)	Définir les attributs pertinents (a_0, a_1, ..., a_n) de Dim_i dans une source de données
DimScheme	Schéma de dimension, en-ligne / hors-ligne (en ligne est la valeur par défaut)
nr_map	Nombre de Mappers
nr_reduce	Nombre de Reducers

4.2 Traitement de Dimensions

ETLMR emploie les primitives de MapReduce : Map, Partition, Combine, et Reduce pour traiter les données. Cependant cet emploi est caché à l'utilisateur qui ne spécifie que les transformations à appliquer aux données et les déclarations des tables de dimension et des tables de fait. Une tâche Map/Reduce lit les données par itération sur les lignes à partir d'un ensemble de données partitionnées. Une ligne est d'abord traitée par Mapper, puis par Partitionner qui détermine le Reducer cible, et ensuite par Combiner qui groupe les valeurs ayant la même clé. Les données sont ensuite écrites dans un fichier intermédiaire (il ya un

fichier pour chaque Reducer). Dans l'étape Reduce, un lecteur de Reducer lit une liste de paires (clé,valeur) à partir d'un fichier intermédiaire et invoque le Reducer pour traiter la liste. Voici les différentes approches pour traiter les données de dimension :

4.3 One Dimension One Task ODOT

Dans cette approche, les tâches Map traitent les données pour toutes les dimensions en appliquant des transformations définies par l'utilisateur et en trouvant les parties pertinentes de la source de données pour chaque dimension. Les données pour une dimension spécifique sont ensuite traitées par une seule tâche Reduce. On nomme cette méthode Une Dimension Une Tâche (ODOT : One Dimension One Task).

L'unité de données qui se déplacent à l'intérieur d'ETLMR est un dictionnaire qui fait correspondre les noms des attributs aux valeurs. Ici, nous appelons cela une **ligne**, par exemple :

ligne = {'url' : 'www.dom0.tl0/p0.html', 'size' : '12553', 'serverversion' : 'SomeServer/1.0', 'downloaddate' : '2011-01-31', 'lastmoddate' : '2011-01-01', 'test' : 'Test001', 'errors' : '7'}

ETLMR lit les lignes à partir des fichiers d'entrée et les transmet aux Mappers. Un Mapper fait *projection* sur les lignes pour élaguer les données inutiles pour chaque dimension et forme des paires (clé,valeur) qui doivent être traitées par les Reducers. Si nous définissons dim_i pour une table de dimension et ses attributs pertinents sont (a_0, a_1 ..., a_n), dans le schéma de la source de données, le Mappeur va générer la sortie comme :
(clé,valeur) = (dim_i.name, $\Pi_{a0, a1, ..., an}$ (ligne)) où *name* représente le nom de la table de dimension. Le Partitionner de MapReduce partitionne la sortie de Map en se basant sur la clé, c'est dim_i.name, telles que les données de dim_i seront soumises à un seul Reducer (voir Figure 34).

Pour optimiser, les valeurs des clés identiques (nom de la table de dimension) sont combinées dans le Combiner avant qu'elles ne soient envoyées aux Reducers tels que le coût de la communication en réseau peut être réduit. Dans le Reducer, une ligne peut subir d'abord des UDFs (User Defined Functions) pour appliquer des éventuelles transformations sur les données arrivées, puis la ligne traitée est insérée dans le magasin de dimension, c.-à-d., la table de dimension dans le DW ou dans un magasin de dimension hors-ligne (décrit plus loin).

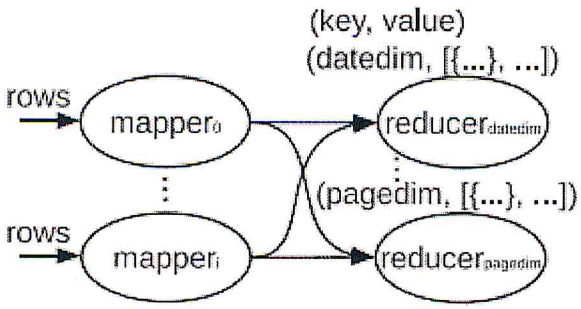


Figure 32 : schéma ODOT [26]

Cette méthode est la méthode la plus fondamentale pour le traitement de dimension où seulement un nombre limité de Reducers peut être utilisé. Par conséquent, son inconvénient est qu'elle n'est pas optimisée pour le cas où certaines dimensions contiennent de grandes quantités de données, à savoir les dimensions de données intensives. [26]

4.4 One Dimension All Task ODAT

Nous décrivons maintenant une autre approche dans laquelle toutes les tâches Reduce traitent les données pour toutes les dimensions. Nous appelons cela Une Dimension Toutes les Tâches (ODAT : One Dimension All Tasks). Dans certains cas, le volume de données d'une dimension est très important. Si nous employons ODOT, la tâche de traitement de données pour cette table de dimension déterminera la performance globale (en supposant que toutes les tâches sont exécutées sur des machines similaires). Nous raffinons ainsi l'ODOT en deux endroits, la partition de la sortie de Map et les fonctions Reduce. Avec ODAT, ETLMR partitionne la sortie de Map en utilisant le partitionnement Round Robin tels que les Reducers reçoivent équitablement plusieurs lignes (voir figure 38). [26]

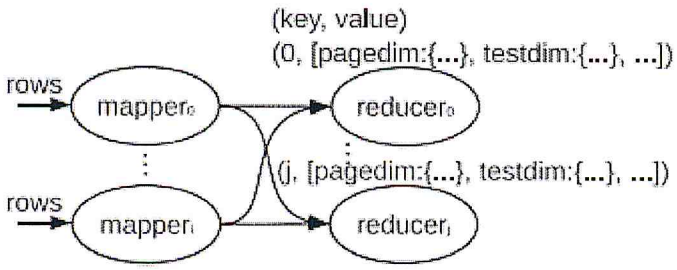


Figure 33 : schéma ODOT [26]

4.5 Dimension en-ligne et hors-lignes

En ODOT et ODAT, les tâches de Map/Reduce interagissent avec les dimensions du DW *en-ligne* directement via des connexions de base de données au moment de l'exécution, la performance est affectée par le DW SGBD extérieur et le coût de la communication avec la base de données. Pour optimiser, le schéma de dimension *hors-ligne* est proposé, dans lequel les tâches n'interagissent pas avec le DW directement, mais avec des dimensions hors-ligne distribuées résidant physiquement dans tous les nœuds. Il a plusieurs caractéristiques et avantages. Tout d'abord, une dimension est divisée en sous-dimensions multiples de plus petite taille, ce qui constitue un gain dans les recherches (lookups) sur les dimensions, en particulier pour les dimensions des données à forte intensité. Deuxièmement, les systèmes de stockage de haute performance peuvent être utilisés pour conserver les données de dimension. Les dimensions sont configurées pour être entièrement ou partiellement mises en cache dans la mémoire principale pour accélérer les recherches lors du traitement des faits. En outre, les dimensions hors-ligne ne nécessitent pas une communication directe avec le DW et les frais généraux (à partir du réseau et le SGBD) sont considérablement réduits. [26]

4.6 Traitement des faits

Le traitement des faits est la deuxième phase en ETLMR, qui consiste à rechercher des clés de dimension, faire l'agrégation sur les mesures (le cas échéant), et le chargement des faits traités dans le DW. De même pour le traitement des dimensions, les définitions et les paramètres des tables de fait sont déclarés dans le fichier de configuration. ETLMR fournit la classe *BulkFactTable* qui supporte le chargement en vrac des faits au DW. Un exemple d'une table de faits:

Testresultsfact = BulkFactTable(name='testresultsfact', keyrefs=['pageId','TestId','DateId'], measures=['erreurs'], bulkloader=UDF PGCOPY, bulksize=5000000).

Les paramètres sont le nom de la table du fait, une liste des clés référençant les tables de dimension, une liste de mesures, la fonction de chargement en vrac, et la taille du vrac à charger. Le chargeur de masse est une UDF qui peut être configurée pour répondre à différents types de SGBD.

Algorithm 3 *process_fact(row)*

Require: A *row* from the input data and the *config*

```
1: facttbls ← the fact tables defined in config
2: for facttbl in facttbls do
3:     dims ← the dimensions referenced by facttbl
4:     for dim in dims do
5:         row[dim.key] ← dim.lookup(row)
6:     rowhandlers ← facttbl.rowhandlers
7:     for handler in rowhandlers do
8:         handler(row)
9:     facttbl.insert(row)
```

Algorithme 3 montre le pseudo-code pour le traitement des faits.

La fonction peut être utilisée comme une fonction Map ou une fonction Reduce. Si aucune agrégation (comme somme, moyenne, ou comptage) n'est nécessaires, la fonction est configurée pour être la fonction Map et l'étape Reduce est omis pour une meilleure performance. Si les agrégations sont nécessaires, la fonction est configurée pour être la fonction Reduce comme les agrégations doivent être calculées à partir de toutes les données. Cette approche est flexible et bonne pour la performance. Ligne 1 récupère les définitions des tables de faits à partir du fichier de configuration, qu'elles sont ensuite traitées de manière séquentielle dans les lignes 2-8. Le traitement consiste en deux opérations majeures:

1) rechercher les clés des tables de dimension référencées (ligne 3-5), et

2) traiter les données de fait utilisant les *rowhandlers*, qui sont des fonctions de transformation définies par l'utilisateur utilisées pour les conversions de type de données, le calcul des mesures, etc. (ligne 6-8).

Ligne 9 invoque la fonction d'insertion pour insérer les données de fait dans le DW. Les données de fait traitées ne sont pas insérées dans la table de fait directement, mais plutôt ajoutées dans un tampon de taille configurable où elles sont conservées temporairement.

Lorsqu'un tampon devient plein, ses données sont chargées dans le DW en utilisant le chargement en vrac. Chaque tâche Map/Reduce a un tampon séparé et un chargeur en vrac telles que les tâches peuvent faire le chargement en vrac en parallèle. [26]

5. Conclusion

Finalement, nous avons retenu MapReduce pour sa simplicité ; le modèle est facile à comprendre et utiliser ; dans un point de vu d'implémentation, c'est facile à utiliser par les programmeurs qui n'ont pas d'expérience avec les systèmes parallèles et distribués, car il cache les détails de la parallélisation, la tolérance aux pannes, l'optimisation de la localité, et l'équilibrage de charge. Aussi, une grande variété de problèmes est facilement exprimable avec MapReduce. MapReduce est utilisé par Google pour la génération des données pour le service de recherche de production de Web de Google, pour le tri, le data mining, l'apprentissage de la machine, et de nombreux autres systèmes.

Partie 3 : Conception et mise en œuvre

Chapitre VI : Conception du système

Chapitre VII : Implémentation du Système

Chapitre VI

Conception du système

Chapitre VI : Conception du système

Ce chapitre s'intéresse à la conception de notre application et au principe de fonctionnement de chacune des méthodes et concepts utilisées pour réaliser une chaine ETL basée sur la logique de MapReduce.

1. Description générale

Dans le but d'obtenir une meilleure performance en termes de temps et d'évolutivité (scalabilité) pour un processus ETL destiné pour le traitement de données massives (Big Data), nous avons retenu l'idée de paralléliser les trois phases du processus à savoir la phase E (Extracting), la phase T (Transforming) et la phase L (Loading). Pour ce faire, nous avons opté pour le paradigme MapReduce largement utilisé pour le calcul parallèle, particulièrement pour les données intensives. Le Framework Apache Hadoop étant une référence pour les plateformes open source, nous l'avons adopté pour mettre en place une plateforme d'intégration 100% parallèle pour faire face au phénomène du Big Data.

Dans ce qui suit, nous allons détailler les modules proposés pour l'intégration basée sur un processus ETL avec le paradigme MapReduce.

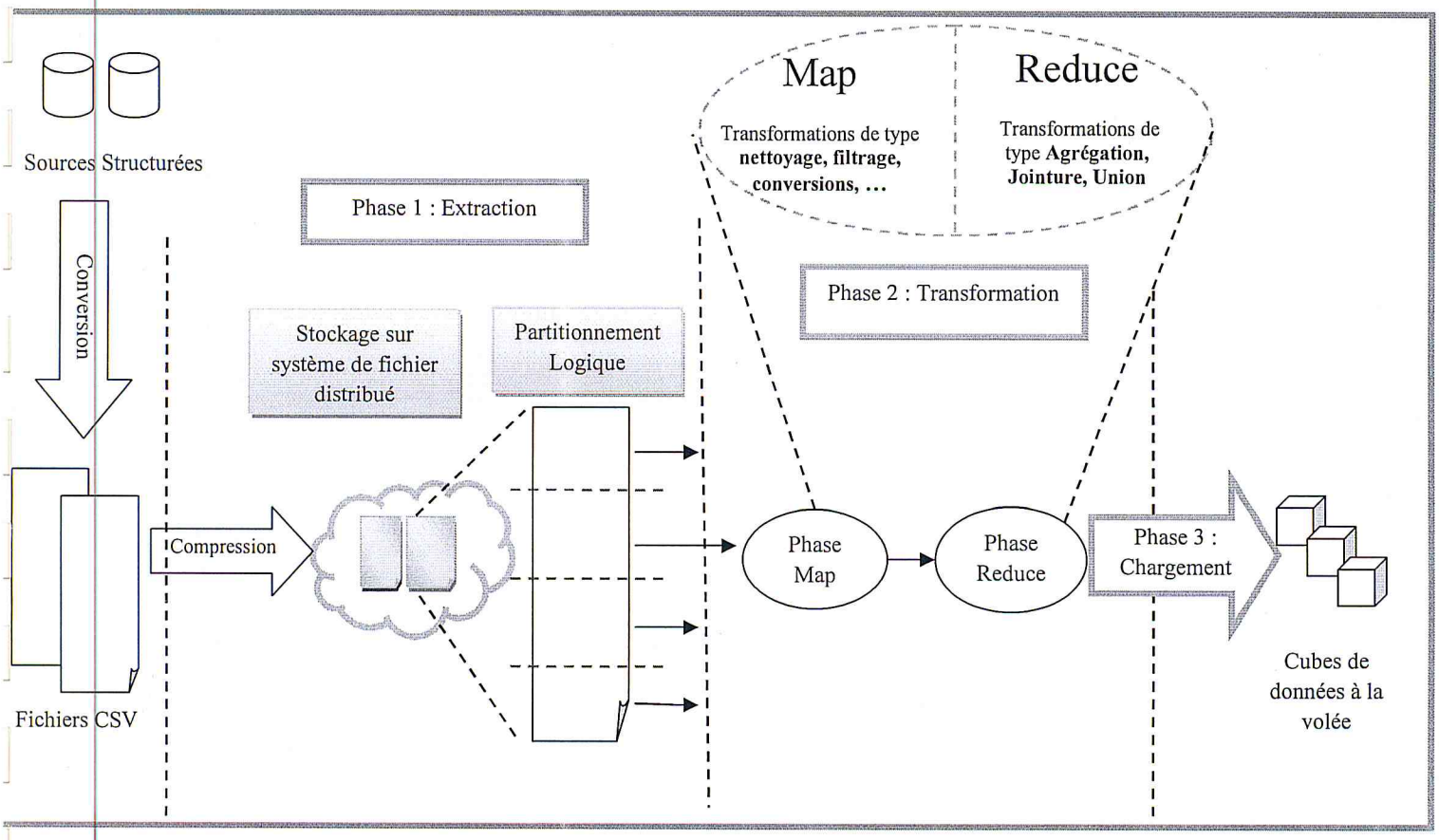


Figure 34 : Architecture globale de notre système

2. Source de données

Les sources de données sont des tables de base de données ou des fichiers de différents formats qui contiennent une collection de données à extraire pour les injecter dans le processus.

Le type de données d'entrées favorable dans notre système est le fichier plat de type CSV ou TEXTE (*.txt, *.csv, etc.), ce type de fichiers est de taille toujours minimale parce qu'il contient du texte sans aucune structuration (gras, italique, soulignage etc.) et aucune donnée structurée (images, vidéos etc.), d'où le choix d'être le principal type des données sources surtout qu'ils sont à l'échelle de Big Data. Aussi sa consultation ne nécessite pas l'installation d'un logiciel à part un simple éditeur de texte disponible sur tous les systèmes actuels. D'autre part, pour donner la capacité à notre plateforme ETL de lire à partir d'autres sources, comme une base de données relationnelle ou une base de données NoSQL, notre système fait l'extraction des données à partir de ces sources puis les convertir dans le format pivot que nous avons adopté (fichier plat). Nous appelons cette opération la « Conversion » des données.

2.1 Fichier plat

Un fichier plat contient des données et le format de ces données. Le format des données peut être spécifié à l'aide d'un séparateur personnalisé, d'une longueur fixe ou par des colonnes que vous définissez avec un nom et un type de données.

```
Matricule,Dateins,Cycle,Specialite,Bourse,Sport
01,2009-09-15,Licence,SI,Null,Oui
02,2009-10-02,Licence,SI,Non,Oui
03,2009-09-08,Master,IC,Non,Non
```

Figure 35 : Exemple d'un fichier plat

2.2 Sources structurées

2.2.1 Base de données NoSQL « MongoDB »

NoSQL désigne une catégorie de systèmes de gestion de base de données (SGBD) qui n'est plus fondée sur l'architecture classique des bases relationnelles. L'unité logique n'y est plus la table, et les données ne sont en général pas manipulées avec SQL.

2.2.2 Base de données relationnelle « MySQL »

Une base de données relationnelle est un stock d'informations décomposées et organisées dans des matrices appelées relations ou tables conformément au modèle de données relationnel. Le contenu de la base de données peut ainsi être synthétisé par des opérations d'algèbre relationnelle telles que l'intersection, la jointure et le produit cartésien. Les informations sont stockées sous forme de groupe de valeurs : « les enregistrements». Un ensemble d'enregistrements relatif à un sujet forme une relation et stocké dans une table. La base de données comporte une ou plusieurs tables.

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur. C'est pour cette raison qu'il fait partit de notre source de données.

Une table SQL pourrait se présenter de la manière suivante :

Nom	Prénom	Âge
DUMONT	Jean	43
PELLERIN	Franck	29
MATTHIEU	Nicolas	51

Figure 37 : Exemple d'une table de base de données relationnel

3. Conversion

Notre système est basé sur des fichiers plats (TXT, CSV). Donc les fichiers structurés doivent passer par une opération de conversion qui consiste à transférer les données d'une source structuré vers un fichier plat à partir d'une requête, et les convertir en séparant les colonnes de l'enregistrement avec un séparateur bien définie pour permettre ensuite la lecture de ces données et valeurs ou d'en spécifier une nouvelle, puis les charger ligne par ligne.

Toute en gardant les noms des colonnes de ces fichiers structurés dans le début du fichier plat. Les sources de données peuvent être une base de données MySQL, Base de données NoSQL « MongoDB ».

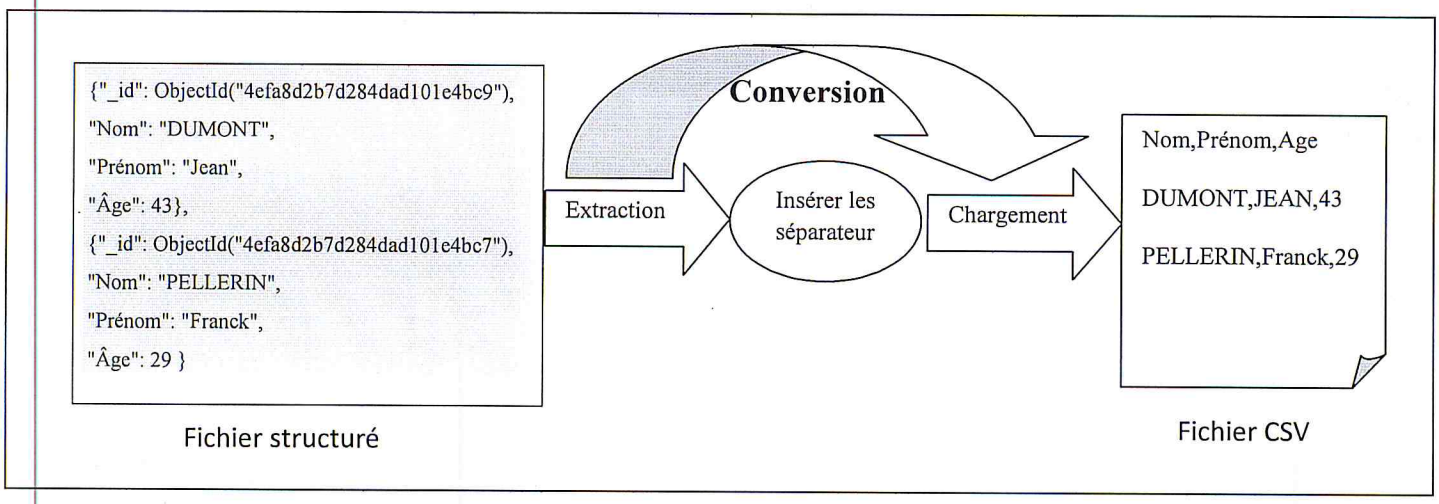


Figure 38 : La tâche de conversion

4. Compression

Comme nous travaillons avec des données volumineuses, il faut s'inquiéter sur le stockage et le temps de transfert de ces données d'un nœud à l'autre pour les traiter.

Pour remédier à ces problèmes, nous avons proposé d'utiliser une méthode de compression pour ces données avant de les traiter qui apporte deux avantages majeurs: elle réduit l'espace nécessaire pour stocker des fichiers, et il accélère le transfert des données à travers le réseau ou à destination ou à partir du disque. Pour traiter les grands volumes de données, ces deux économies peuvent être significatives.

Le stockage des données compressées dans le système de fichier distribué l'HDFS permet à l'allocation de votre matériel d'aller plus loin puisque les données compressées sont souvent 25% de la taille des données d'origine. En outre, puisque les jobs MapReduce sont presque toujours lié au E/S (Entrées/Sorties), le stockage des données compressées signifie qu'il ya moins d'E/S globales à faire, implique des jobs plus rapides. Il y a deux inconvénients à cela cependant : certains formats de compression ne peuvent pas être fractionnés (splittable) pour le traitement parallèle, et d'autres sont assez lent à la décompression que les jobs deviennent liés au processeur, en éliminant vos gains sur les E/S.

Dans un environnement parallèle, nous devons utiliser une méthode de compression qui est splittable.

5. Stockage dans un système de fichier distribué

Une fois les fichiers sources sont prêts, nous les stockons dans un système de fichier distribué qui nous permet d'accéder à une arborescence de fichiers ne résidant pas sur la machine locale mais sur les autres machines du réseau (les serveurs de fichiers) tout en utilisant la même syntaxe que pour accéder au système de fichiers local. Il s'agit de diviser les fichiers sources et les distribuer à touts les nœuds du cluster de manière fiable pour assurer l'accessibilité à toutes les machines de cluster.

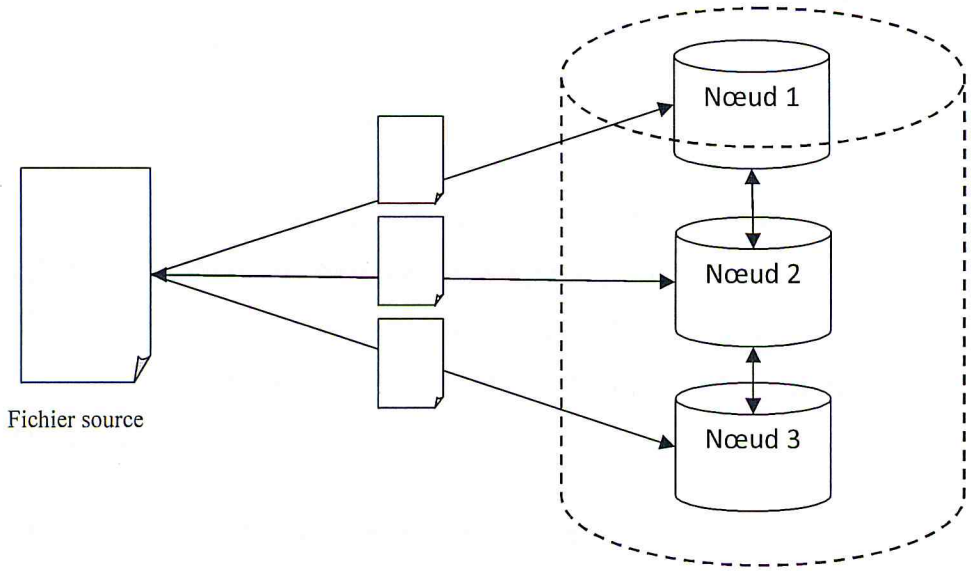


Figure 39 : Concept de système de fichier distribué

6. Extraction

L'extraction est la 1ère étape dans le processus ETL et le point d'entré des données à notre système. Dans cette phase notre système lit un fichier plat. A cet effet la tâche d'extraction doit récupérer les données de fichier source et les soumettre ligne par ligne à notre système pour les traiter, comme nous voulons extraire ces données d'une manière parallèle, et les pris en charge avec plusieurs tâches, cela nécessite un partitionnement de ces données.

- NOT NULL (NN) qui contrôle l'existence de valeurs NULL dans les attributs (date inscription, spécialité, cycle).
- conversion d'une date format américaine (date inscription) en format Européen.
- L'extraction de l'année à partir de la date d'inscription.
- L'agrégation qui consiste à calculer les effectifs des étudiants par cycle, spécialité et année.

Les données sont distribuées dans le fichier source d'une manière que les lignes qui contient des valeurs NULL dans l'un de ces attributs (date inscription, spécialité, cycle) sont localisés dans le début de fichier avec pourcentage de 60% de la totalité des données du fichier, dans ce cas la si nous appliquons la technique de partitionnement simple qui est basé sur la division de la taille des partitions d'une façon égale, nous allons obtenir un déséquilibrage de la charge de transformation au niveau des Mappers, ou les Mappers chargés de traiter le début de fichiers va prendre moins de temps par rapport aux Mappers chargé de traiter le reste à cause de la déférence de nombre d'opérations exécutées par chacun des Mappers, car nous trouvons une seule opération sur les lignes localiser au niveau de début du fichier source, lorsque le Mapper trouve une valeur NULL dans les attributs définies précédemment, il passe à l'autre ligne sans faire les autres transformations, par contre 4 opérations sur les lignes qui reste c à d les lignes qui ne vérifient pas la condition de l'existence d'une valeur NULL. En outre la phase Reduce attends la terminaison de tous les Mapper avant de commencer. Cela laisse un impact négatif sur la performance en termes de temps.

Pour cela nous avons proposé une technique de partitionnement intelligent basé sur le facteur de distribution de la charge de transformation sur les Mappers.

Principe :

a. Echantillon de fichier source

Dans un premier temps nous prendrons un échantillon de 5% de la totalité des données du fichier source aléatoirement, et pour chaque ligne de l'échantillon nous calculons le nombre d'opérations suivant les transformations prévus dans la phase Map, et on génère un fichier nommé « **InputSampler** » qui contient la position de la ligne on octet et le nombre d'opérations appropriés

InputSampler.txt

```
24      1
89      1
102     2
187     1
205     1
222     1
270     1
301     4
450     2
 .
 .
 .
9872973     3
9889869     4
9932873     2
```

Ce qui nous donne ce graphe on fonction de nombre d'opérations et la position :

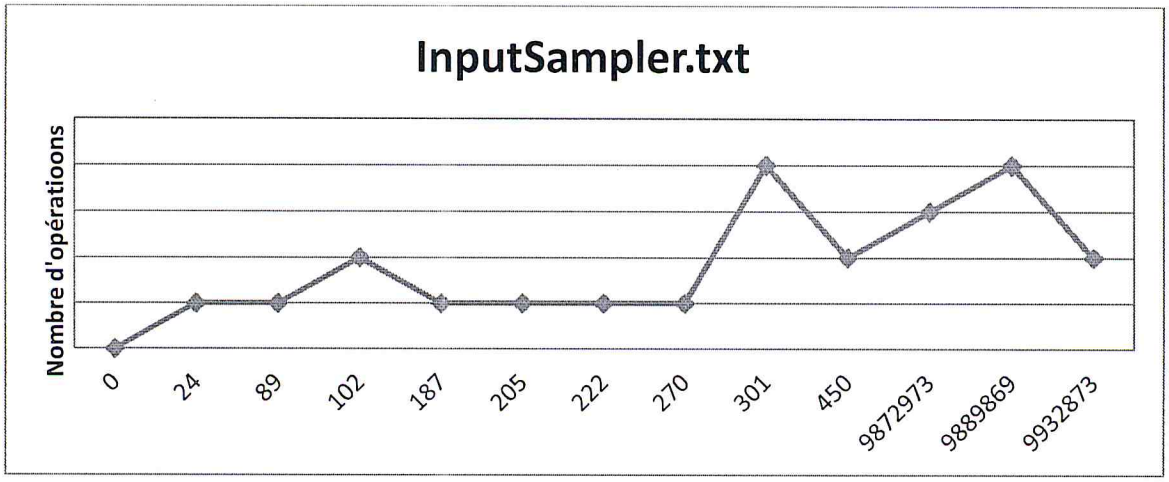


Figure 40 : Représentation graphique de l'InputSampler

b. Interpolation

Ensuite, nous avons besoin de faire une interpolation avec ces points, cela nous permet de construire une fonction à partir de ces points, en utilisant la méthode d'interpolation Spline car elle est la méthode préférable à l'interpolation polynomiale parce que l'erreur d'interpolation est très minime.

c. Calcule d'intégrale

Après l'interpolation on peut calculer l'intégrale de la fonction produit par cette dernière « Interpolation» dans l'intervalle [0, Taille de fichier source] avec la méthode trapèze.

Le principe de cette méthode d'intégration est analogue à celui de l'intégration par une série de rectangles. Dans cette méthode on cherche à augmenter la précision du calcul tout en évitant la dissymétrie qui apparaît dans la définition de l'intégration par rectangles. Nous allons donc ici tabuler f(x) en n+1 points $f_0, f_1 \ldots f_n$ dans l'intervalle d'intégration $[x_0, x_n]$. Ensuite nous utilisons les valeurs calculées précédemment par l'interpolation pour remplacer les arcs x_i, x_{i+1} par leurs cordes.

L'intégration va alors se faire en sommant l'aire des trapèzes, soit :

$$I = \sum_{i=0}^{n-1} \frac{f_{i+1} + f_i}{2} (x_{i+1} - x_i)$$

d. Identifier les partitions logiques

Après le calcule de l'intégrale, nous le devisons sur le nombre de Mappers et nous obtenons le résultat **R**, ensuite nous cherchons la position x1 tel que l'intégrale dans l'intervalle [0, x1] est égale à **R,** puis nous allons à la position x2 tel que l'intégrale dans la l'intervalle [x1, x2] est égale à R , nous réputons la procédure jusqu'à la dernier position dans le fichier source.

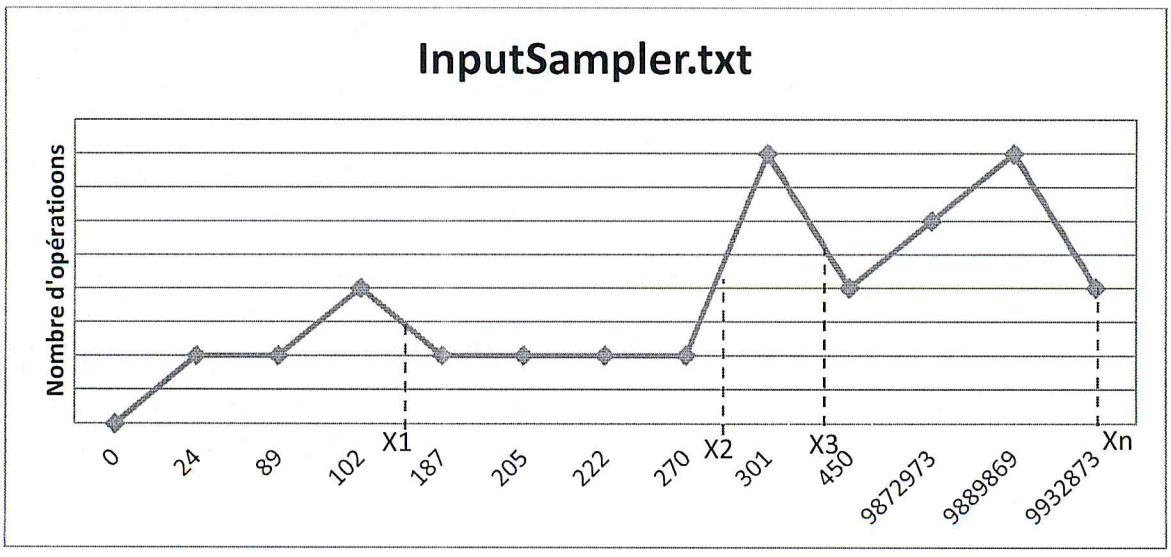


Figure 41 : Identification des positions des partitions

D'après ces positions nous définissons les partitions logiques comme suit :

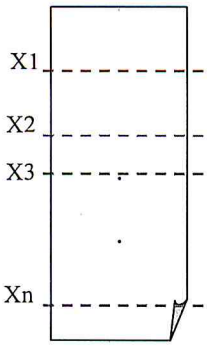


Figure 42 : Définition des partitions logiques

De cette manière on assure une distribution équilibré de la charge de transformation sur les Mappers.

On exécute le job de prélèvement en premier et ensuite on lance le vrai job de L'ETL qui va prendre moins de temps que les autres jobs de L'ETL qui utilise les techniques de partitionnement simple (simple, round robin).

Remarque : **cette technique de partitionnement peut ne pas être idéale lorsque la distribution des données est assez irrégulière.**

6.2 Manière de soumettre les données aux Mappers

Dans toutes les techniques de partitionnement précédentes nous nous sommes basé sur deux manières de soumettre les données aux Mappers :

- ✓ **Par nombre de lignes** : qui consiste à fournir N ligne(s) aux Mappers.
- ✓ **Par la taille de données** : où les Mappers sont alimentés à chaque fois par une quantité bien définie de données.

7. Transformations

Les transformations sont une série de règles à appliquer sur les données extraites à partir de la source pour préparer et consolider les données finales qui seront chargées dans la destination. Elles servent à reformater, fusionner, agréger et nettoyer les données pour les normaliser et se débarrasser des valeurs aberrantes. La transformation est souvent l'étape d'ETL la plus complexe et la plus couteuses en termes de temps de traitement. Son optimisation est un objectif principal.

Les transformations selon leur endroit dans le processus MapReduce sont classées en deux :

7.1 Transformations de Map

Elles sont les transformations qui s'exécutent directement sur les lignes lues à partir du fichier source une fois arrivées aux Mapper. Les lignes sont représentées sous forme de paires (clé,valeur). Le Mapper ne s'intéresse pas à la clé (qui est la position de la ligne dans le fichier, c'est ce qu'on appelle byte offset[2]), mais plutôt à la valeur qui constitue la ligne lue à partir de la source. On prend la ligne, on obtient les valeurs des colonnes par éclatement de cette ligne en se basant sur le délimiteur. On applique la série de transformations sur ces valeurs obtenues, puis on forme de nouveau la paire (clé,valeur) et on l'envoie au Reducer si nécessaire (cas d'agrégation). Le Map peut bien entendu changer la clé en la remplaçant par l'une des valeurs obtenues par exemple, tout dépond du besoin.

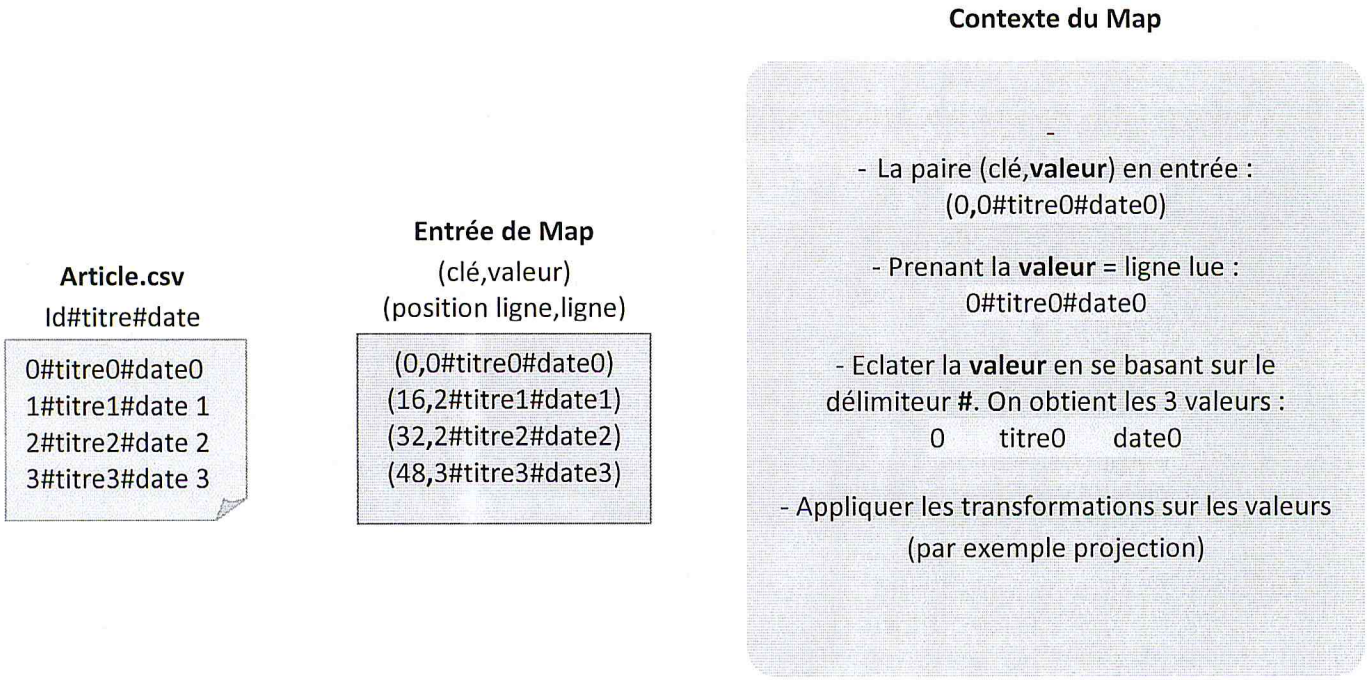


Figure 43 : Transformation de Map (Projection)

7.1.1 La projection : Sert à choisir un sous-ensemble de colonnes, à partir de la source, qui vont subir une série de transformations avant qu'elles soient chargées dans la destination. C'est le SELECT dans une syntaxe SQL.

7.1.2 La restriction : Au contraire de la projection qui selecte des colonnes, la restriction select les lignes satisfaisant des conditions logiques. C'est le WHERE dans SQL.

[2] Byte offset ou le décalage d'octets, généralement utilisé pour indexer dans une chaîne ou un fichier, est le nombre d'octets à partir de zéro. Par exemple, dans la chaîne "Demain le dépôt du mémoire", le décalage d'octet de "Demain" est 0, de «le» est le 7, "dépôt" est 10, et "mémoire" est de 19.

7.1.3 Commutation jour/mois : Faite la conversion d'une date américaine à une date européenne et vice-versa.

7.1.4 Remplacement : Si une valeur est trouvée dans une colonne spécifique, la remplacer par une autre. Comme par exemple : si 'dzd' est trouvée, la remplacer par un vide '' ou si 'true' est trouvée, la remplacé par un 1, dans le cas où la colonne source est de type chaine de caractères et la colonne destination on veut qu'elle soit de type entier.

7.1.5 Concaténation : Cette transformation effectue la concaténation des colonnes, tel que la concaténation des trois colonnes : *jour*, *mois* et *année* dans une seul colonne *date*.

7.1.6 Eclatement : Cette transformation fait l'éclatement d'une colonne donnée en deux ou plusieurs colonnes par un délimiteur bien défini. Par exemple éclater la valeur « 10 – 19 » de la colonne *Température inférieure et supérieure* en deux nouvelles colonnes : *Température supérieure* ayant la valeur 10 et *Température inférieure* ayant la valeur *19* utilisant le délimiteur «–».

7.1.7 Formule : Calculer le résultat d'une formule mathématique, par exemple : Etant données 3 colonnes : A, B et C dans la source. Créant une nouvelle colonne D dans la destination dont la valeur est la formule : ((A*C)/B)*0.75.

7.1.8 Dérivation de la date : On fournit une transformation qui retourne une partie de la date. Par exemple : extraire seulement le mois et année à partir de la colonne *date* puis créer les deux colonnes engendrées : *mois* et *année*.

7.2 Transformations de Reduce

Les données sorties de la phase Map, avant qu'elles soient reçues par le Reducer, sont combinées de telle sorte que toutes les valeurs de la même clé sont regroupées ensemble dans une même paire : c'est l'opération du Shuffle. Les Reducers reçoivent ainsi des paires sous la forme (clé,ensemble de valeur regroupées), il itère ensuite sur ces valeurs pour appliquer des transformations dites « d'agrégation ».

7.2.1 L'agrégation : Sert à regrouper toutes les lignes ayant la même valeur d'une colonne donnée (ou plusieurs colonnes) avec l'une des mesures d'agrégation à savoir : le comptage, la somme, la moyenne, le maximum et le minimum. Par exemple, calculer l'effectif des étudiants par Cycle, ceci revient à donner une liste réduite contenant le nombre des étudiants

de chaque Cycle : Licence, Master et Doctorat. C'est l'équivalent de la clause GROUP BY avec respectivement les fonctions COUNT, SUM, AVG, MAX et MIN de SQL.

Bien que cette transformation est d'agrégation, sa réalisation dans un framework MapReduce est dispersée dans les deux phases Map et Reduce. Parce que c'est à l'étape Map qu'on spécifie les colonnes à agréger. Voici le scénario de la réalisation :

A l'étape Map, on passe la valeur (ou les valeurs) de la colonne (ou les colonnes) qu'on veut agréger comme clé, et les valeurs des autres colonnes concaténées comme valeur dans la paire (clé,valeur). L'opération de Shuffle va regrouper les valeurs de la même clé dans une parie unique permettant au Reducer ensuite d'appliquer l'une des fonctions d'agrégation, dans notre cas, c'est le *comptage* des étudiants. La figure ci-dessous illustre les étapes de la réalisation de la transformation « agrégation » :

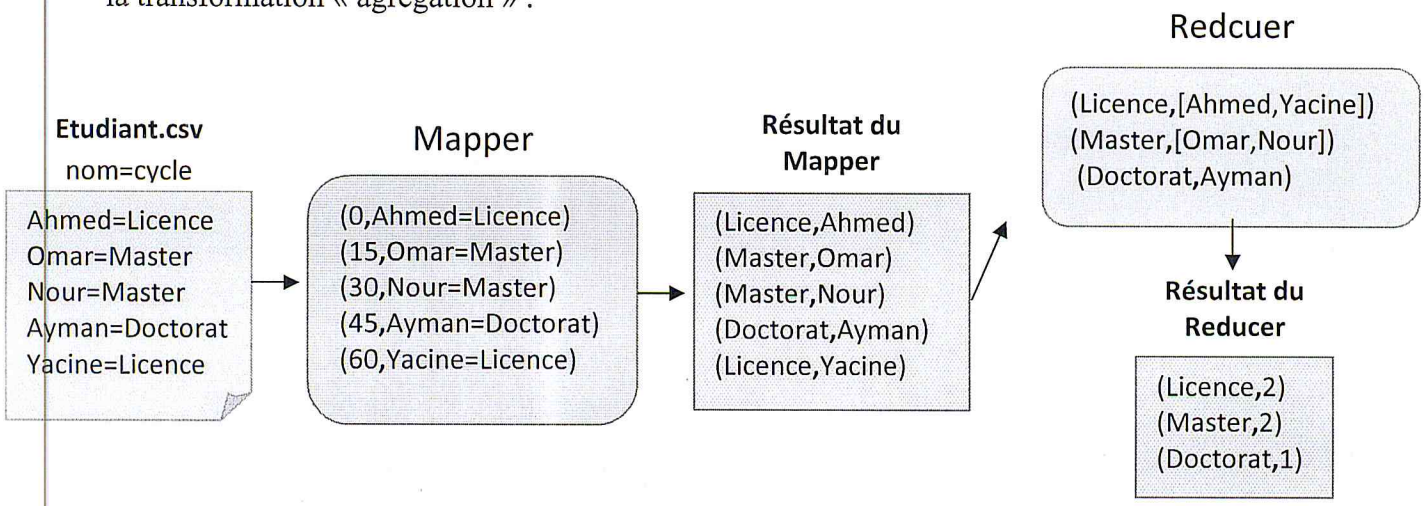


Figure 44 : Transformation de Reducer (Agrégation)

7.2.2 La restriction des agrégations : Elle porte le même effet que la restriction de la phase Map à moins que ça soit sur le résultat de l'agrégation. Comme : donner l'effectif des étudiants par Cycle à condition que cet effectif soit supérieur à 160 étudiants, par exemple. C'est le HAVING dans le langage SQL.

7.2.3 Jointure : Cette fameuse opération dans les bases de données relationnelles nous permet de combiner deux tables en se basant sur une colonne commune à chacune des deux. Dans notre plateforme MapReduce, il s'agit de combiner les données provenant de deux fichiers sources (qui contiennent des données ayant la forme d'une table relationnelle, c.-à-d. des lignes de valeurs séparées par un délimiteur). Nous avons besoins d'un Mapper pour chaque source. Chaque Mapper doit tenir la (valeur de la) colonne, de la source correspondante, sur laquelle sera basée la jointure. Il envoie (la valeur de) cette colonne comme clé et le reste des

(valeurs des) colonnes comme valeur. Le Reducer combine les valeurs en entrée qui sont toutes associées à la même clé (colonne de la jointure). Un exemple clarifie l'opération :

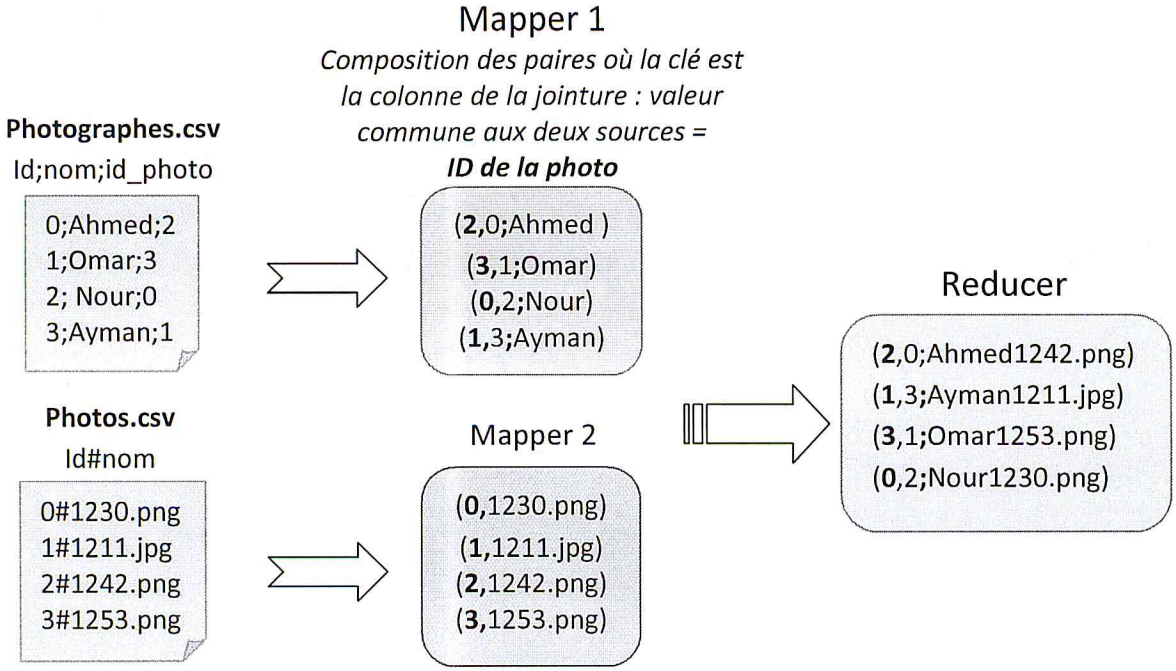


Figure 45 : Transformation de Reducer (Jointure)

Il faut noter que si on utilise plusieurs Reducers, pour garantir la parallèlisation, les résultats de l'agrégation (effectifs des étudiants par Cycle par exemple) seront dispersés dans plusieurs fichiers de résultat (un résultat par Reducer). Pour regrouper ces résultats, nous proposons d'exécuter un autre processus MapReduce à la fin qui prend les résultats dispersés et les agrège.

Une autre solution sert à affecter chaque groupe, par exemple : Licence, Master et Doctorat, à un Reducer unique, cela nous donne des résultats correctes (sans dispersion) mais le fait de limiter le nombre de Reducers au nombre de ces groupes risque de diminuer le parallélisme.

7.2.4 Fusion

Il s'agit de fusionner deux sources ayant la même structure de lignes de données, c.-à-d. le même nombre et type de colonnes. Cette opération est simple, le framework MapReduce considère toutes les sources d'entrée comme une seule source, tout ce qu'on doit faire alors est d'utiliser des Mappers et des Reducers qui ne font que recevoir les paires (clé,valeur) et les envoyer sans aucune modification (ils sont appelés Mapper Identité et Reducer Identité). Si on veut fusionner des sources dont les colonnes d'une source sont incluses dans les colonnes de l'autre source, on devra appliquer la transformation de la projection pour sélectionner

seulement les colonnes concernées par la fusion. Mais le Mapper est inconscient de la source de la ligne qu'il reçoit, dans ce cas, il peut compter le nombre des colonnes pour savoir sur quelle source il doit exécuter la projection. Dans le cas où le nombre de colonnes de chaque source est le même, et on veut appliquer des transformations sur l'une des sources avant de la fusionner avec l'autre source, le Mapper ne peut pas savoir, dans ce cas, la source de la paire (clé,valeur) reçue pour décider s'il applique les transformations ou pas. Une solution sert à spécifier au Mapper, au préalable, quelle source il va prendre en charge, ainsi, il peut envoyer le nom de la source comme clé. Le Reducer, à partir de la clé de la paire reçue, il peut savoir s'il s'agit d'une ligne de la source qui doit subir les transformations nécessaire avant la fusion.

8. Chargement

La dernière phase dans la chaine ETL c'est l'opération du chargement. Elle fait le chargement des données, ayant subi une série de transformations, dans la destination. Chaque Reducer (ou éventuellement le Mapper) produit un fichier contenant son résultat de travail stocké dans le système de fichier. Le système de fichier se charge ensuite de fusionner toutes les parties dans un seul fichier constituant le résultat final et de le copier sur le disque local dans l'emplacement spécifié par l'utilisateur. C'est un fichier texte contenant des lignes de valeurs séparées par un délimiteur aussi spécifié par l'utilisateur. Ce dernier peut indiquer s'il veut compresser le résultat.

9. Compression des résultats

Nous avons parlé sur les types des fichiers compressés possibles en entrée des Mappers. Nous parlons maintenant de la compression des résultats produits par les Reducers. Si l'utilisateur veut que les résultats soient compressés, il spécifie un type de compression parmi les trois types : Gzip, Bzip2 et LZO.

10. Conclusion

Nous avons décrit dans ce chapitre la conception de notre plateforme spécifique à la chaine ETL qui fonctionne dans un environnement de big data distribué et parallèle en se basant sur le paradigme MapReduce, avec les éventuelles méthodes conceptuelles qui servent à optimiser l'opérabilité de notre plateforme pour gagner en performances : moins de temps avec moins de ressources pour plus de productivité. Dans le chapitre suivant, nous allons parler sur l'implémentation d'une telle plateforme à savoir les différents outils utilisés pour sa mise en œuvre.

Chapitre VII : Implémentation du système

Après avoir achevé la conception du système, nous aboutissons à l'avant dernière étape de développement qui est la phase d'implémentation, se traduisant par une implémentation des différents modules figurant dans l'architecture du système.

1. Implémentation

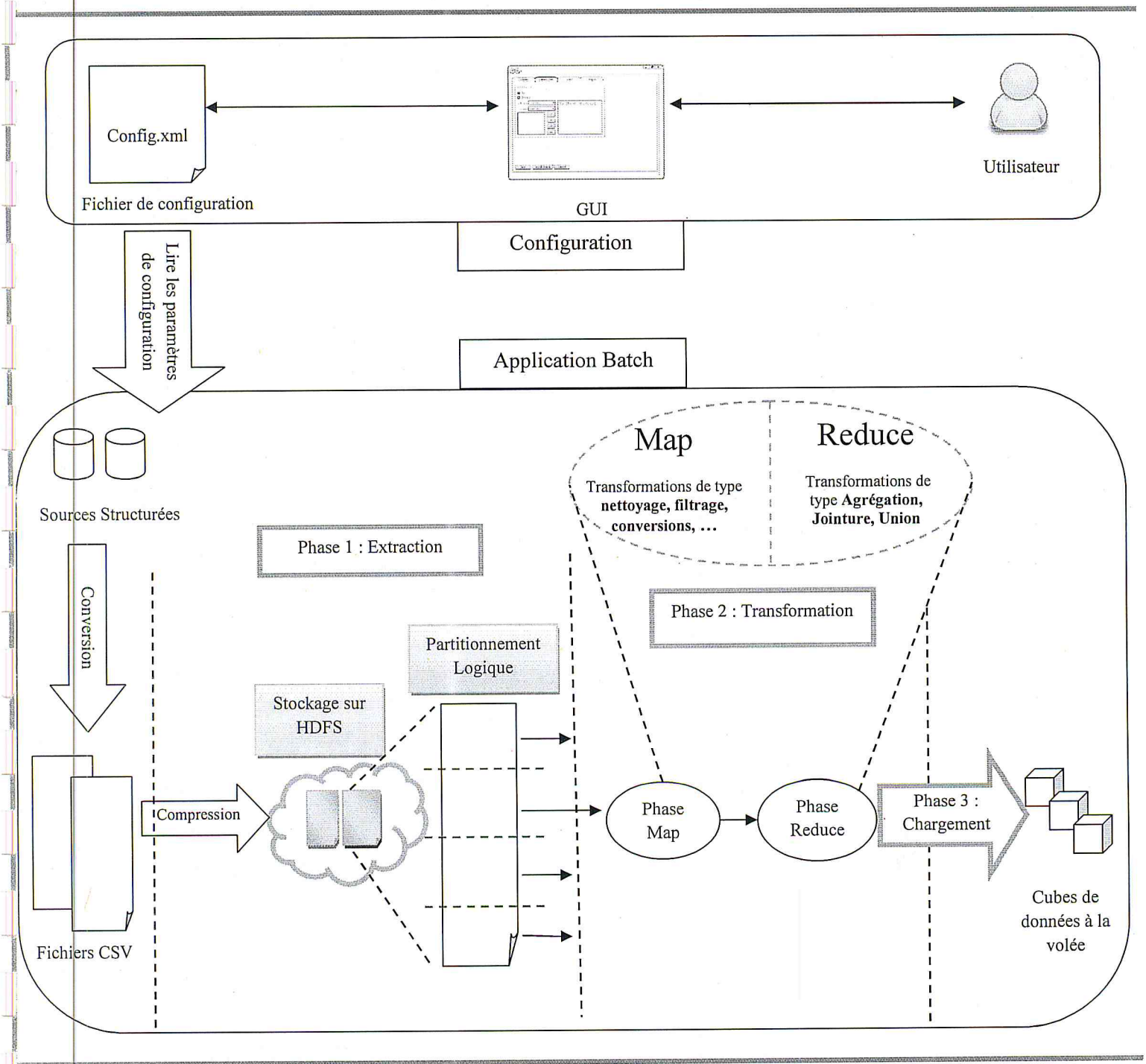


Figure 46 : Architecture détaillée de notre système

Chapitre VII : Implémentation du système

Nous avons implémenté la logique de MapReduce avec le framework de Hadoop qui se charge de la distribution des données avec son système de fichier distribué HDFS et la parallélisation du calcule, donc on est intervenu pour implémenter nos méthodes que nous avons défini dans le chapitre précédant.

1. 1 Fichier de configuration

Notre application est de type batch (par lots), ça veut dire que l'enchaînement de ses opérations se fait automatiquement lors de l'exécution sans l'intervention de l'utilisateur. Il faut configurer toutes les attributs et les opérations de l'application à l'avance, c.-à-d avant son lancement, nous disposons d'un fichier spécial dit : de configuration, nommé **config.xml** contenant les propriétés et les réglages à imposer lors de l'exécution de notre application.

C'est à l'utilisateur de notre plateforme de configurer les instances de l'application à l'aide d'une interface utilisateur graphique GUI (des captures d'écran sont disponible au partie de présentation de l'interface d'utilisateur) en choisissant les différentes propriétés à respecter et les opérations à appliquer (les transformations, les méthodes de lecture et de partitionnement des données, le type de compression, le chemin du fichier source ainsi que d'autres propriétés à respecter).

Le fichier de configuration est sauvegardé après validation. Une fois l'application lancée elle va lire les paramètres de ce fichier de configuration et les sauvegarder dans la mémoire RAM pour les utiliser au cours de l'exécution.

Le format de notre fichier de configuration étant XML. Nous avons choisi XML pour les avantages :

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML.
- Auto descriptif et extensible.
- Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques.
- Universalité et portabilité : les différents jeux de caractères sont pris en compte.
- Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP.

- Intégrabilité : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML).
- Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

Ainsi, XML est souvent utilisé dans l'échange de données.

Par exemple si on travaille sur un fichier source (scolarité) pour calculer les effectifs par année d'inscription, spécialité et cycle avant leur chargement dans l'entrepôt de données, quatre transformations sont prévues dans cet exemple :

- NOT NULL (NN) qui contrôle l'existence de valeurs NULL dans les attributs (date inscription, spécialité, cycle).
- conversion d'une date format américaine (date inscription) en format Européen.
- L'extraction de l'année à partir de la date d'inscription.
- L'agrégation qui consiste à calculer les effectifs des étudiants par cycle, spécialité et année.

Le fichier de configuration généré pour cet exemple est le suivant :

```
1  < ?xml version ="1.0" encoding="UTF-8">
2  <config>
3  <concept>
4   <input-path>/home/hduser/scolarite.csv</input-path>
5   <output-path>/home/hduser/</output-path>
6   <!--Task : Transformation -->
7   <transformations>
8     <transformation type="SwitchDate" from="usa" to="fr" get="year">
9       <field>1<field>
10    <transformation>
11   <transformation type="restriction"> < !—WHERE-->
12      <field type="ne" value ="NULL" >1</field>
13      <field type="ne" value ="NULL" >2</field>
14      <field type="ne" value ="NULL" >3</field>
15    <transformation>
16   <transformation type="projection"> <!-- SELECT -->
17       <field>0</field>
18       <field>1</field>
19       <field>2</field>
20    </transformation>
21  </transformations>
22
23
```

```
24  <aggregation type="1">
25    <!--1: COUNT by pk fields
26        2: SUM field by pk fields
27        3: AVG field by pk fields
28        4: MIN field by pk fields
29        5: MAX field by pk fields       -->
30        <field>1</field>
31        <groupBy>1,2,3</groupBy>
32  </aggregation>
33 </concept>
34 </config>
35
```

1.2 Implémentation de l'Extraction

Pour la phase de l'extraction on est intervenu sur les classes suivantes :

InputFormat : Cette classe hérite les fonctionnalités et les propriétés de la classe FileInputFormat qui a été fourni par Hadoop, à travers de cette classe on a défini comment les fichiers d'entrée sont séparés logiquement en un seul ou plusieurs InputSplit suivant les méthodes de partitionnement que nous avons détaillé dans la conception.

InputSplit : qui décrit une unité de travail qui comprend une seule tâche de map dans un programme MapReduce, on a personnalisé cette classe suivant nos méthodes de partitionnement.

RecordReader : La classe InputSplit définit une partition logique, mais ne décrit pas comment y accéder. Avec la classe RecordReader nous avons implémenté la manière de la lecture de données sources définit par le InputSplit et le convertit en paires (clé, valeur) adaptées pour la lecture par le mappeur. L'instance de RecordReader est définie par la classe InputFormat.

HDFS : on a créé cette classe pour les commandes liées au Hadoop, par exemple : start-all.sh, stop-all.sh (démarrer /arrêter tous le daemons) et HDFS comme : ls, mv, copyFrmLocal, etc. (pour la manipulation des fichiers).

1.2.1 Compression

Hadoop accepte trois types de compression a chacun ses avantages et inconvénients en termes de temps et d'espace de compression : Gzip, Bzip2 et LZO. Les différents outils ont des caractéristiques de compression très différentes. Gzip est un compresseur d'usage général, et se trouve au milieu du compromis espace/temps. Bzip2 compresse plus efficacement que Gzip, mais elle est plus lente. La vitesse de décompression de Bzip2 est plus rapide que la vitesse de compression, mais il est encore plus lent que les autres formats. LZO, d'autre part,

optimise la vitesse : il est plus rapide que Gzip (ou tout autre outil de compression ou décompression), mais compresse légèrement moins efficace [29].

Pour atteindre des performances maximales et des prestations, nous devons utiliser la méthode de compression LZO et Bzip2 qui sont caractérisés par la rapidité de compression et de décompression, ainsi que la taille de fichier compressé est beaucoup plus petit que les autres formats de compression, en outre il nous permet de diviser les fichiers compressé en morceaux et distribuées à tous les nœuds du cluster de manière fiable comme s'il s'agit d'un fichier texte normal.

1.2.2 Partitionnement logique et la technique de lecture

Avant de citer les différentes techniques de la lecture parallèle par les Mappers, nous expliquons la logique du partitionnement de Hadoop. Hadoop ne fait pas un partitionnement physique où le fichier source sera divisé en petits morceaux de données, chaque morceau est lu par un Mapper, mais plutôt un partitionnement logique. Hadoop possède une fonction appelée **getSplit** qui retourne une liste des identificateurs des zones de lecture de chaque Mapper, chaque zone est identifiée et représentée sous la forme d'un objet appelé **InputSplit** qui est défini par le triplet (S, D, F) ou S est le chemin du ficher source, D le décalage d'octet à partir duquel le Mapper commence sa lecture dans le fichier (appelé souvent start offset), F le nombre d'octets qui devraient être lus par le Mapper chargé (c.-à-d. commencer à partir de la position D jusqu'à la position D+F). Par la suite, chaque Mapper est nourri par un lecteur appelé **RecordReader** qui lit les données de la zone correspondante, ligne par ligne, et les convertit en paires (clé, valeur). Cette opération est répétée jusqu'à ce que la totalité d'**InputSplit** soit consommée. Chaque invocation de **RecordReader** conduit à un autre appel à la fonction map() du Mapper comme illustre cet algorithme :

```
1  Debut
2      Nouveau Recordreader ( inputSplit(S, D, F) ) ;
3      Tant que( Recordreader.CleValeurSuivant() <> Nul )
4      Faire
5          map ( Recordreader.CleCurrent(), Recordreader.ValeurCurrent() ) ;
6      Fait.
7      Nettoyage(Recordreader) ;
8  Fin.
```

Dans ce contexte nous utilisons cette logique pour implémenter des nouvelles méthodes de partitionnement logique :

a. Partitionnement simple

Il s'agit de partitionner logiquement le fichier source en morceaux dont la taille est approximativement égale à (taille fichier source/nombre de Mappers). Donc au niveau de getSplit() on génère des inputSplits avec des tailles égales suivant le nombre des Mappers. Chaque Mapper lit sa propre partition défini par l'objet inputSplit.

```
1  Debut
2      List Partitions = nul ;
3      Taille_fichier = taille (fichier source) ;
4      Chemin_fichier = Chemin (fichier source) ;
5      Lire(nb_Mappers) ;
6      Taille_partition = Taille_fichier / nb_Mappers ;
7      Octets_ restants=Taille_fichier ;
8
9      Tant que( ( Octets_restants / Taille_partition ) > 1)
10     Faire
11         Partitions.ajouter(inputSplit( Chemin_fichier, Taille_fichier – Octets_restants, Taille_partitions)) ;
12         Octets_restants = Octets_restants – Taille_partition ;
13     Fait.
14
15     Si ( Octets_restants <>0)
16     Alors
17         Partitions.ajouter( inputSplit( Chemin_fichier, Taille_fichier – Octets_restants, Octets_restants)) ;
18     Fin.
19 Fin.
```

b. Partitionnement Round Robin

Pour appliquer cette technique avec la logique de partitionnement de MapReduce, nous avons personnalisé l'objet inputSplit(S, D, F, I) ou **S** est le chemin de ficher source, **D** le décalage d'octet à partir duquel le Mapper commence sa lecture dans le fichier (souvent appelé start offset), dans les limites de la split qui lui a été affectée, **F** le nombre d'octets qui devrait être lu par le Mapper et **I** étant l'index de partitionnement.

Ensuite on génère à l'aide de la fonction getSplit() **n** inputSplits(S, D, F, I) ayant la même taille de fichier source avec des indexes **I** allant de 1 jusqu'à **n** et chacun de ces inputSplit étant ensuite affecté à un Mapper, et ce dernier parcourt tous le fichiers et prend les lignes qui remplissent la condition (**I ={numéro de ligne modulo nombre de Mappers}**).

Voila l'étape de création des partitionnes logique :

```
1  Debut
2      List Partitions = nul ;
3      Taille_fichier = taille (fichier source) ;
4      Chemin_fichier = Chemin (fichier source) ;
5      Lire(nb_Mappers) ;
6
7      Pour( i=0, i < nb_Mappers, i++ )
8       Faire
9          Partitions.ajouter(inputSplit( Chemin_fichier, 0, Taille_fichier, i )) ;
10     Fait.
11 Fin.
```

Et dans la partie de la lecture au niveau du Mapper :

```
1  Debut
2      Entier Index_fichier = inputSplit.Index ;
3      Lire(nb_Mappers) ;
4      Entier Position = inputSplit.Debut ;
5      Entier Fin_fichier = Position + inputSplit.Longeur ;
6      Fichier Fichier_source = inputSplit.Source ;
7      Entier i = 1 ;
8      Tant que ( position < fin_fichier)
9       Faire
10         Taille_ligne = Lire_Taille_ligne(Fichier_source, Position) ;
11         Si ( i mod nbr_Mapper = Index_fichier)
12         Alors
13            Lire(Fichier_source, Posisiton ) ;
14         Fin ;
15         i = i + 1 ;
16         Position = Position + Taille_ligne ;
17     Fait.
18 Fin.
```

c. Partitionnement Intelligent

Cette technique est basé sur un le facteur de la charge de transformation au niveau des Mappers. Nous générons les partitions logiques avec l'algorithme suivant :

```
1  Debut
2      List Partitions = nul ; // liste des partitions logique.
3      Lire(inputSampler, X) ; // Lire les positions.
4      Lire(inputSampler, Y) ; // Lire les nombres d'opération.
5      Lire(nb_Mappers) ; // lire le nombre de Mappers.
6      Tableaux [nb_Mappers] Position ; // tableaux des positions de partitions.
7      Taille_fichier = taille (fichier source) ; // La taille de fichier source.
8      Chemin_fichier = Chemin (fichier source) ; // récupérer le chemin de fichier.
9      Octets_ restants = Taille_fichier ;
10     Entier j = 0 ;
11     Fonction F = Interpolation.Interpoler(X, Y) ; // Inter
12     Resultat = F.intégrer ( x[0], x[x.Longueur ] ) ; // calcule d'intégrale
13     Partition_intégrale = Resultat / nb_Mappers ;
14     Position [0] = integrer.Posistion (X [0], X [X.longueur], Partition_Intégrale) ;
15     Pour ( k=0, k < nb_Mappers, k++ )
16     Faire
17         Posistion [k] = integrer.Posistion (Position [k-1], X [X.longueur], Partition_Intégrale) ;
18     Fait ;
19     Taille_partition = Position [0] ;
20     Tant que( ( Octets_restants / Taille_partition ) > 1)
21     Faire
22         Partitions.ajouter(inputSplit( Chemin_fichier, Taille_fichier – Octets_restants, Taille_partitions)) ;
23         Octets_restants = Octets_restants – Taille_partition ;
24         J = J +1 ;
25         Taille_partition = Position [J] – Position [J-1]
26     Fait
27     Si ( Octets_restants <>0)
28     Alors
29         Partitions.ajouter( inputSplit( Chemin_fichier, Taille_fichier – Octets_restants, Octets_restants)) ;
30     Fin ;
31 Fin.
```

1.3 Implémentation de la Transformation

Hadoop est un framework MapReduce bas niveau non orienté-métier ; il a été conçu pour distribuer des tâches sur des machines pour qu'elles travaillent en parallèle avec une gestion distribuée des fichiers. Notre travail était de l'adapter pour la réalisation des processus ETL complets. Ce qui nous a mis devant plusieurs défis, entre autres, le manque d'un outil d'expression des requêtes similaires à SQL, ceci a engendré des vrais problèmes quand on arrive à la réalisation des opérations clés comme les sélections, les restrictions, les jointures et les agrégations. Plusieurs solutions open-source ont été mises en place pour pallier ces limitations venant étendre le framework Hadoop. On parle de Hive qui facilite l'interrogation des données utilisant un langage similaire au SQL appelé HiveQL. On parle de Pig, une plateforme haut-niveau qui offre un langage de script en Java avec une notation similaire à

celle de SQL. On parle d'Impala, qui rend possible d'utiliser SQL avec Hadoop pour interagir avec les sources. [28]

Notre objectif était de descendre dans la couche et fournir les interactions nécessaires pour interroger les données avec Hadoop et Java directement sans avoir besoins d'un intermédiaire.

En utilisant le fichier de configuration décrit plus haut, l'utilisateur final, à l'aide d'une interface utilisateur simple, sera en mesure de spécifier les transformations souhaitées. Parmi lesquelles des transformations de projection, de restriction, de jointure et d'agrégation qui sont connues dans les bases de données relationnelles telle que MySQL.

Il y a une classe Transformation qui contient une méthode pour chaque type de Transformation avec les paramètres nécessaires. Bien que nous ayons fourni les transformations les plus utilisées dans un ETL, la liste n'est pas exhaustive, d'autres types de transformation peuvent être ajoutés au fur et à mesures.

1.4 Implémentation de Chargement

La dernière phase dans la chaine ETL c'est l'opération du chargement. Elle fait le chargement des données, ayant subi une série de transformations et de consolidations, dans la destination. Chaque Reducer (ou éventuellement le Mapper si zéro Reducer est utilisé) produit un fichier texte contenant son résultat de travail stocké dans le système de fichier HDFS. L'HDFS se chargera ensuite de fusionner toutes les parties dans un seul fichier constituant le résultat final, et de le copier sur le disque local dans l'emplacement spécifié par l'utilisateur dans l'interface. C'est un fichier texte plat contenant des lignes de valeurs séparées par un délimiteur aussi spécifié à l'aide de l'interface. L'utilisateur peut indiquer un type de compression s'il souhaite compresser son résultat.

Le format natif des résultats produits par le framework Hadoop est le fichier texte plat. Notre plateforme offre autres manières de stocker les données, c'est de sauvegarder les résultats dans une base de données relationnelle MySQL ou dans une base de données NoSQL MongoDB dans laquelle un entrepôt de données peut résider. L'utilisateur configure, à travers l'interface utilisateur, la connexion avec le server MySQL ou le serveur MongoDB installé sur sa machine en spécifiant le host, le port, l'utilisateur, le mot de passe, le nom de la base de données et le nom de la table (collection dans MongoDB) qui va recevoir les résultats. L'utilisateur est responsable de la validité de la connexion et l'existence de la base de données cible.

1.5 Adaptation du framework Hadoop avec le traitement de Big Data [30]

Nous avons utilisé les paramètres de configuration de Hadoop pour s'adapter nos besoins :

1.5.1 Compression de sortie de Mappers

La sortie du Mapper peut être compressée avant qu'elle soit sauvegardée sur le disque pour accélérer l'écriture sur disque, utiliser moins d'espace, et réduire la quantité de données à transférer au Reducer à partir des Mappers. Par défaut, la sortie du Mapper n'est pas compressée, mais comme nous travaillons sur le big data et exécutons nos jobs sur des clusters, il est fort recommandé d'utiliser la compression pour une meilleure performance.

1.5.2 Exécution spéculative

Lorsque une tâche (Map ou Reduce) fonctionne très lentement (en raison de dégradation du matérielle ou d'une mauvaise configuration) que prévu. Le JobTracker exécute une autre tâche équivalente à cette tâche comme un backup (sauvegarde) sur un autre nœud. Ceci est connu sous le nome l'Exécution spéculative. La sortie de la tâche qui termine la première sera prise et l'autre tâche sera tuée. Par défaut, l'exécution spéculative est activée, mais dans les gros jobs où le temps moyen d'achèvement de la tâche est important (> 1 heure) en raison des complexes et grands calculs et la nécessité au haut débit, l'exécution spéculative doit être mise false.

1.5.3 Mémoire tampon pour le tri

La sortie générée par le Mapper est écrite dans une mémoire tampon circulaire pour trier la sortie, la taille de cette mémoire est de 100Mo par défaut. Cependant pour les grands jobs (les jobs dans lesquels la sortie du Mapper est très grande), la taille devrait être augmentée en gardant à l'esprit que cette augmentation doit être en fonction de la mémoire disponible au niveau du nœud.

1.5.4 Facteur de tri

MapReduce garantit que l'entrée de chaque Reducer est triée par clé. On rappelle que le processus par lequel le système effectue le tri et transfère les sorties de la map pour les Reducers est connu comme le Shuffle. La phase de tri devrait être appelée après la phase de fusion. Cette phase commence lorsque toutes les Mappers ont été achevés et leur production a été copiée. Les sorties du Mapper sont fusionnés en conservant leur ordre de tri.

Le facteur de tri est le nombre maximum de flux à fusionner à la fois lors du tri des fichiers, la valeur par défaut de cette propriété est 10. Cependant pour les grands jobs (où la sortie du Mapper est très grande et le nombre de Mappers est également grands), la valeur de cette propriété doit être augmentée.

1.5.5 Réutilisation de JVM

Le framework Hadoop nous permet de spécifier le nombre maximum de tâches à exécuter pour un job donné pour chaque JVM sur un TaskTracker. Par exemple : la même JVM peut être utilisé pour toutes les tâches d'un job.

La surcharge de la création JVM pour chaque tâche est d'environ 1 seconde. Donc, pour les tâches qui vivent pendant quelques secondes ou quelques minutes et ont une longue initialisation, cette valeur peut être augmentée pour gagner en performance, mais dans notre cas où les tâches prennent beaucoup de temps et s'exécutent dans un environnement big data, nous avons configuré cette propriété pour exécuter une seule tâche au niveau de JVM.

1.5.6 Parallélisation de copier

Le nombre de threads utilisés pour copier les sorties du Mapper pour la phase Reduce doit être augmenté en gardant à l'esprit que cela augmentera l'utilisation totale du CPU.

1.5.7 Nombre de Reducers

Le nombre de Reducers utilisé varie d'un Job à l'autre. La valeur par défaut est 1. Cela signifie que la sortie de toutes les tâches de la phase Map sera envoyée à un seul Reducer. Le réglage du nombre optimal de Reducers est une science elle-même. Dans la documentation JobConf il est recommandé d'utiliser cette formule [27] :

1.75 * (**Numbre Nœud**) * (**le nombre maximum de Reducers qui peut êtres s'exécuter simultanément sur un nœud**)

Pour calculer le nombre maximum de Reducers qui peut êtres exécuter simultanément sur un nœud il faut que l'utilisateur indique la taille de mémoire disponible sur les nœuds et la taille de la mémoire allouée pour chaque tâche. La valeur de ce dernier est égale à {La taille de mémoire disponible sur les nœuds / la taille de la mémoire allouée pour chaque tâche}

2. Outils de développement

a. Apache Maven

Est un outil logiciel libre pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier. L'objectif recherché est comparable au système Make sous Unix : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.

Il est semblable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML. Maven est géré par l'organisation Apache Software Foundation. Précédemment Maven était une branche de l'organisation Jakarta Project.

Maven utilise un paradigme connu sous le nom de Project Object Model (POM[3]) afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches pré-définies, comme la compilation de code Java ou encore sa modularisation.

Un élément clé et relativement spécifique de Maven est son aptitude à fonctionner en réseau. Une des motivations historiques de cet outil est de fournir un moyen de synchroniser des projets indépendants : publication standardisée d'information, distribution automatique de modules jar. Ainsi en version de base, *Maven* peut dynamiquement télécharger du matériel sur des *dépôts* logiciels connus. Il propose ainsi la synchronisation transparente de modules nécessaires [W07].

b. NetBeans

NetBeans Est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML.

Conçu en Java, NetBeans est disponible sous Windows, Linux Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme.

NetBeans inclut le support complet Maven depuis 6.7, y compris Maven 3 support en 7.0 [W07].

c. Ubuntu

[3] Un modèle d'objet du projet ou POM est l'unité fondamentale du travail dans Maven. Il s'agit d'un fichier XML qui contient des informations sur le projet et les détails de configuration utilisés par Maven pour construire le projet. Il contient des valeurs par défaut pour la plupart des projets

Est une distribution GNU/Linux qui réunit stabilité et convivialité. Elle s'adresse aussi bien aux particuliers qu'aux professionnels, débutants ou confirmés qui souhaitent disposer d'un système d'exploitation libre et sécurisé.

« Ubuntu » est un ancien mot africain qui signifie « Humanité ». Ubuntu signifie également « Je suis ce que je suis grâce à ce que nous sommes tous ». La distribution Ubuntu apporte l'esprit Ubuntu au monde logiciel [W08].

d. Hadoop

Nous avons utilisé le framework Hadoop et son système de fichier distribué HDFS, Plus de détails sur cette Framework et ses composants, sont disponibles au annexe.

e. Java

Le langage **Java** est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au *SunWorld*.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et Frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java. [W07]

3. Présentation de l'interface utilisateur

Dans ce qui suit nous allons présenter les différentes interfaces réalisées qui permettent à l'utilisateur de spécifier la configuration qui définit les transformations souhaitées ainsi que les paramètres de l'extraction et le chargement.

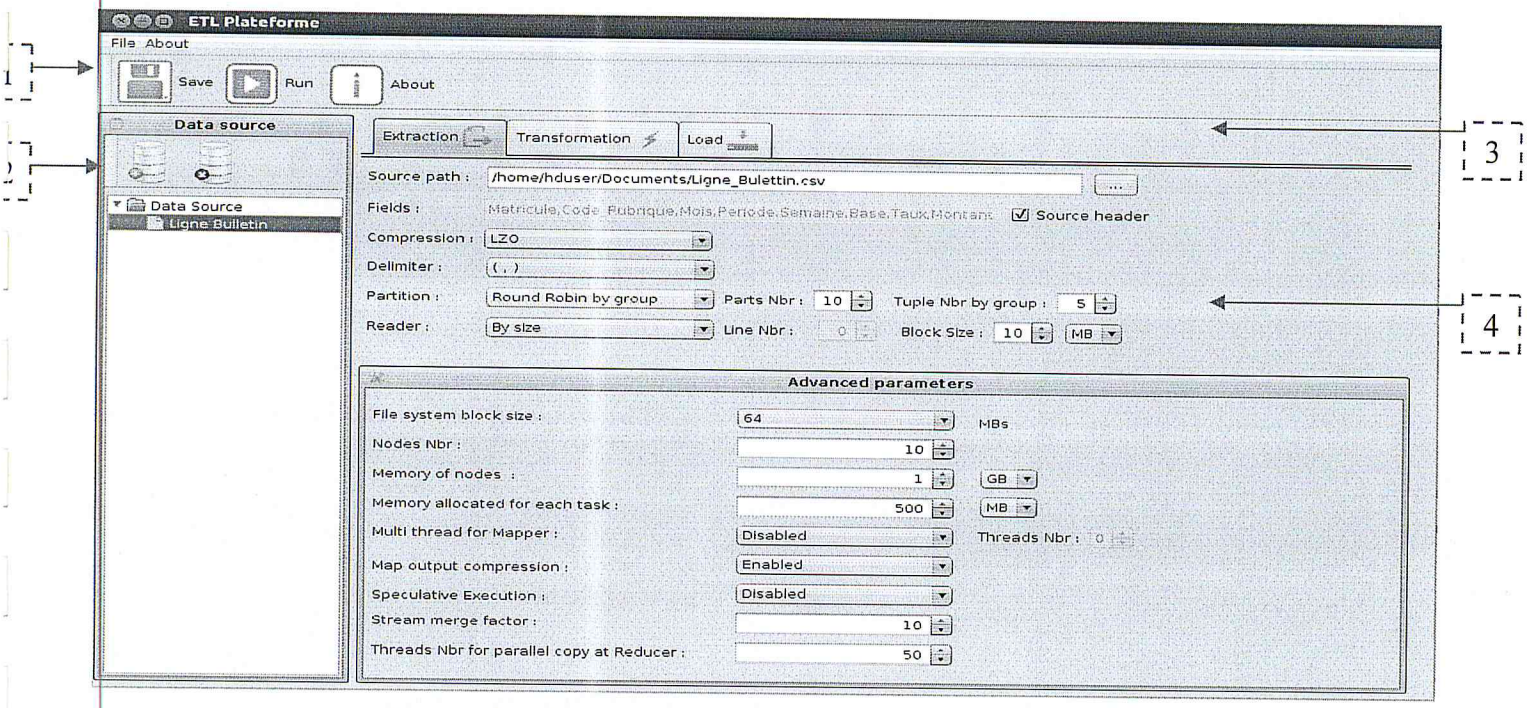


Figure 47 : L'interface principale de notre application

Dans cette interface nous avons 4 parties principales :

Partie 1 : C'est la barre de menu principale ou l'utilisateur peut lancer le processus ETL et sauvegarder les configurations dans le fichier de configuration de l'application qui doit être lu au moment de lancement du processus.

Partie 2 : Dans cette partie l'utilisateur peut ajouter ou supprimer des sources de données.

Partie 3 : C'est la zone du paramétrage de la phase de l'extraction ou l'utilisateur doit spécifie :

- le chemin de fichier sources.
- Les colonnes de fichier source.
- Type de compression de fichier source avant l'extraction.
- Méthode de partitionnement.

- La manier de la lecture des tuples de fichier sources.

Partie 4 : Les paramètres avancés de notre application qui permettent à l'utilisateur de changer la configuration par défaut pour les phases Map et Reduce pour obtenir une bonne performance du processus ETL, ces paramètres comprennent :

- La taille de block HDFS.
- Le nombre de nœuds du cluster.
- La taille de mémoire du nœud.
- La mémoire allouée pour chaque tâche.
- Le multithread pour le Mapper.
- Compression de sorties de Mappers.
- L'exécution spéculative.
- Le facteur de tri.
- Le nombre de threads utilisés pour copier les sorties du Mapper pour la phase Reduce.

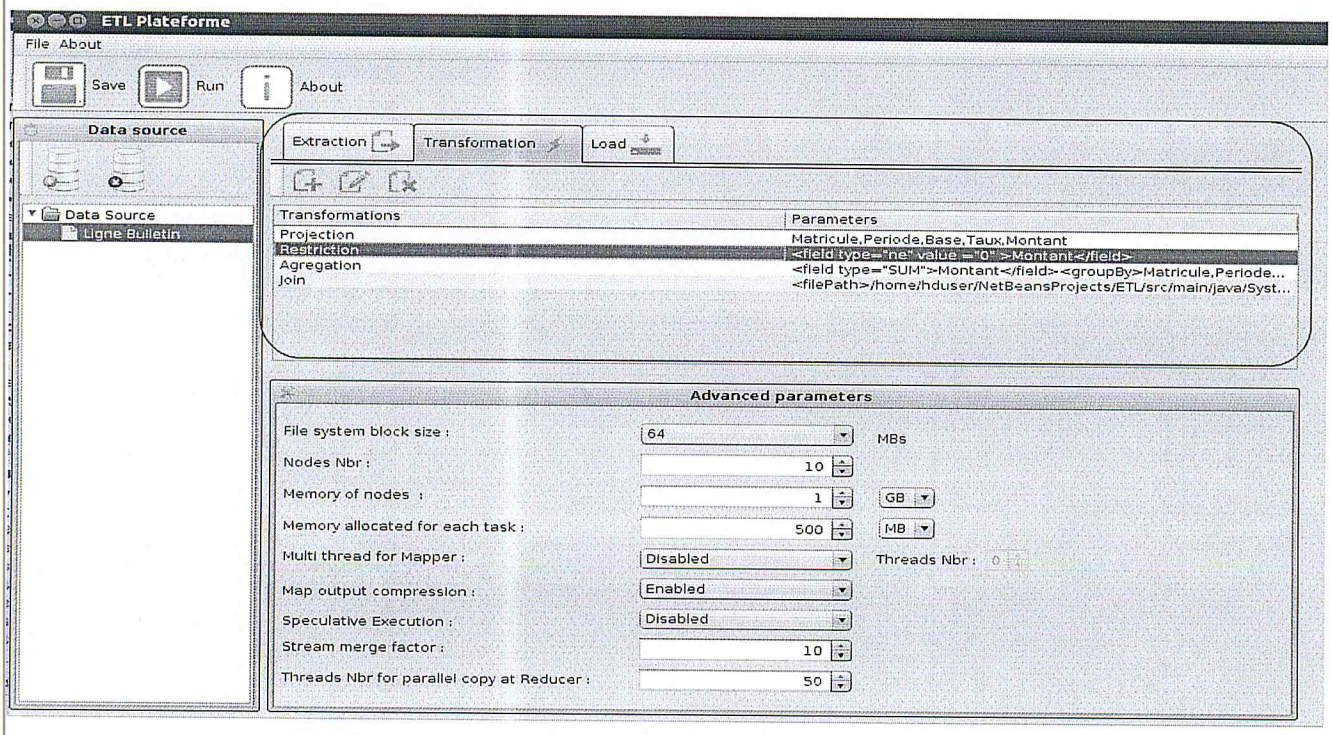


Figure 48 : Paramétrage de la phase transformation

Comme illustre la Figure 2, L'utilisateur peut ajouter ou modifier, supprimer les transformations souhaitées dans l'onglet de transformation.

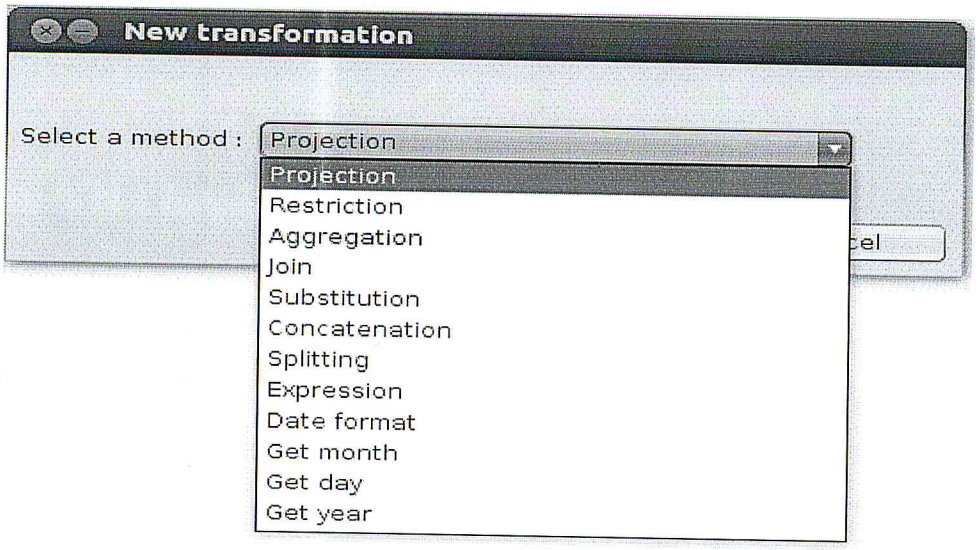


Figure 49 : Ajoute d'une nouvelle transformation

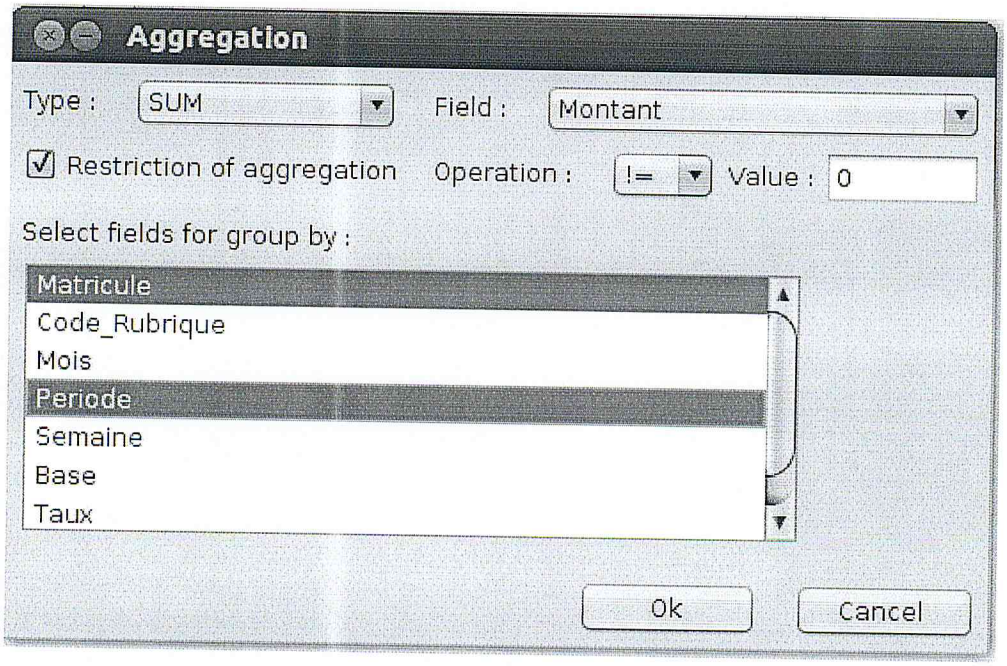


Figure 50 : Interface de configuration de l'agrégation

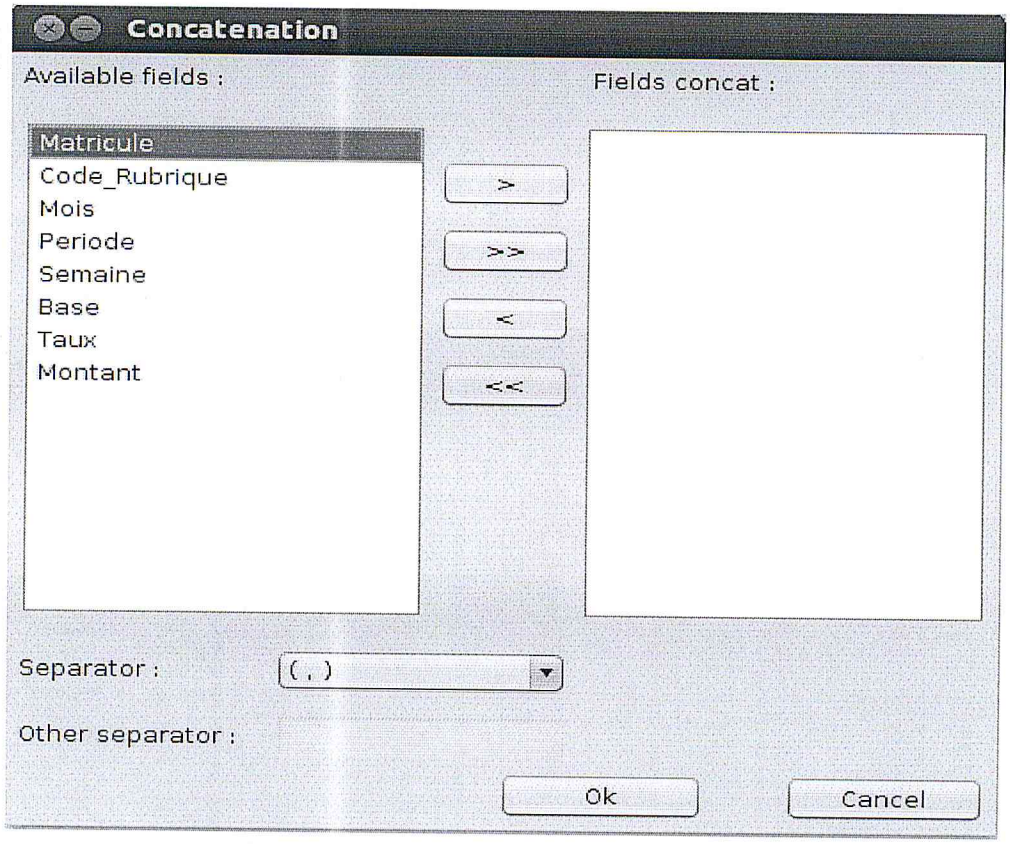


Figure 51 : Interface de configuration de la concaténation

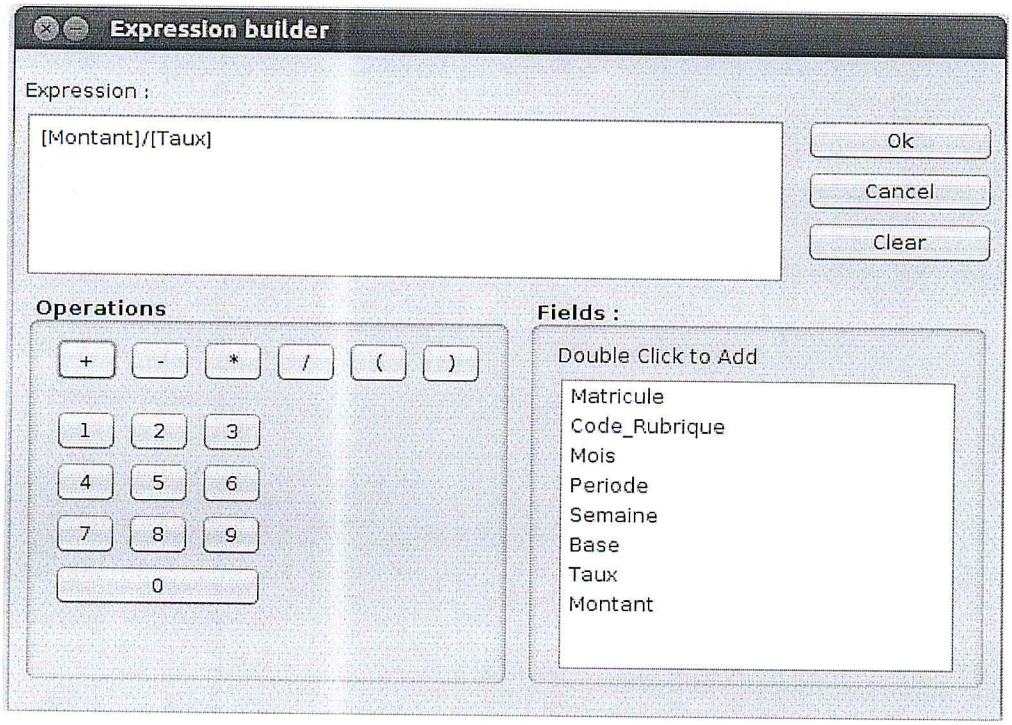


Figure 52 : Interface de construction d'une formule

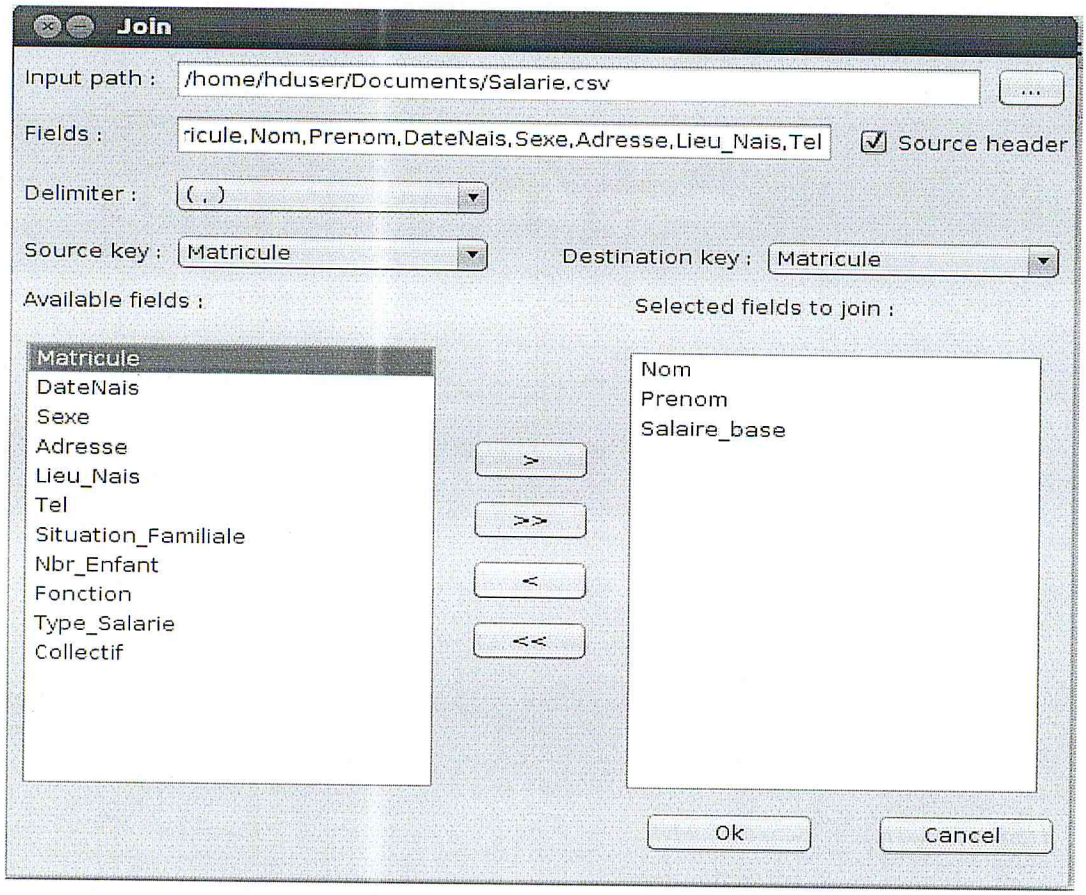


Figure 53 : Interface de configuration de la jointure

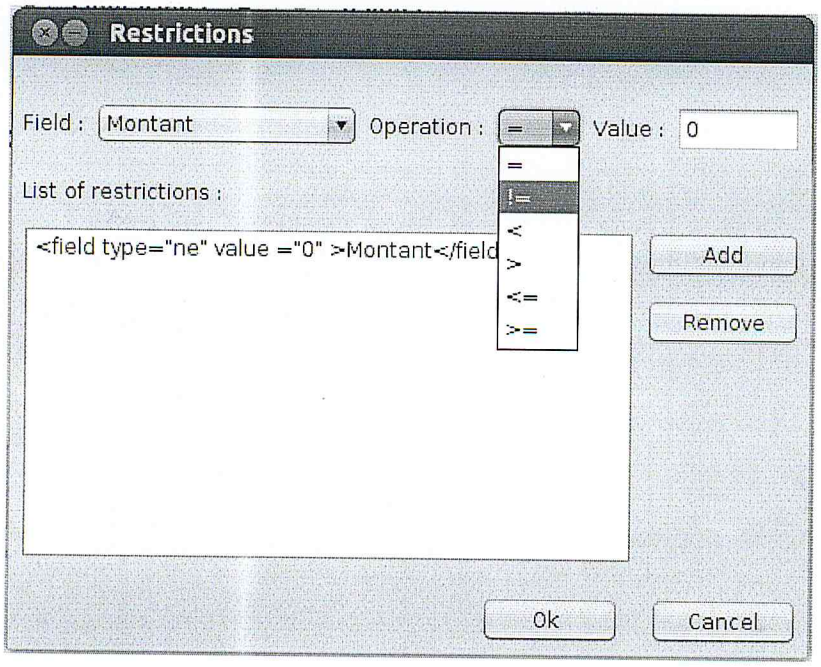


Figure 54 : Interface de configuration de la restriction

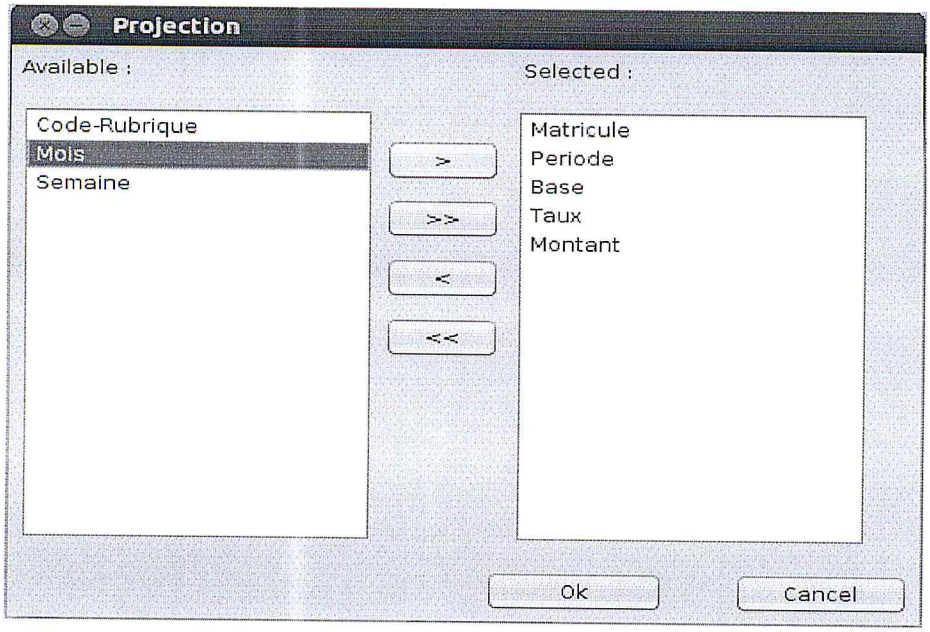


Figure 55 : Interface de configuration de la projection

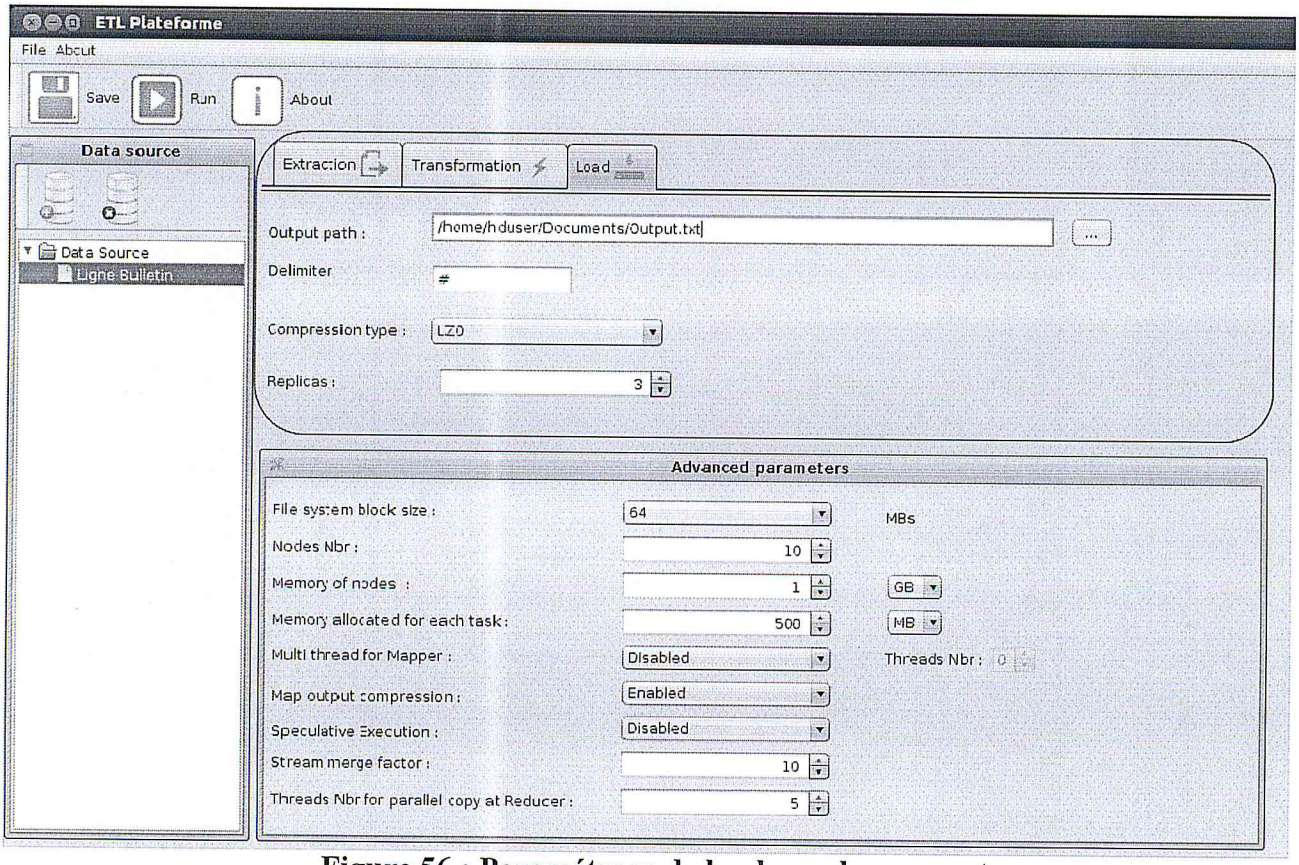


Figure 56 : Paramétrage de la phase chargement

Dans cet onglet l'utilisateur doit spécifier :

- Le chemin de fichier de sortie.
- Le séparateur utilisé pour séparer les champs de fichier de sortie.
- La compression de fichiers de sortie.
- Le Facteur de réplication HDFS.

4. Conclusion

Nous avons décrit dans ce chapitre la partie de l'implémentation ainsi que les différents outils utilisés pour la réalisation de notre plateforme tout en préservant autant que possible la structure définie par l'architecture globale de notre système, puis nous avons présenté l'interface de notre application qui permet à l'utilisateur d'utiliser notre plateforme d'une manière simple et facile.

Conclusion générale

et

Perspectives

Conclusion générale et perspectives

1. Conclusion générale

L'objectif assigné à notre travail était de mettre en œuvre une plateforme ETL pour l'intégration des données à partir des sources de données hétérogènes dans un cube ou un entrepôt de données selon le paradigme de calcul parallèle MapReduce pour traiter parallèlement, au sein d'un cluster de serveurs, des données massives, connues aujourd'hui sous le nom de Big Data. Notre contribution était essentiellement focalisée sur la parallélisation et l'optimisation des performances de chacune des trois opérations d'ETL à savoir l'extraction, la transformation et le chargement des données. Nous avons adopté une implémentation MapReduce qui a fait ses épreuves dans le traitement parallèle des grandes quantités des données dans des clusters de matériels standards, elle a été utilisée largement par des entreprises de grand renom tel que Microsoft, IBM et Amazon. C'est le framework Hadoop.

Dans la phase E (Extracting), nous avons travaillé sur deux préoccupations à savoir (1) la diversité des sources et (2) le partitionnement ainsi que la distribution des données de façon équitable sur les serveurs du cluster pour une utilisation meilleure des ressources matérielles disponibles. Notre plateforme ETL est basée sur le framework Hadoop, ce dernier adopte le fichier plat (csv ou txt) comme format pivot. Les données dans un fichier plat doivent avoir une structuration similaire à celle des tables relationnelles, c.-à-d., des valeurs séparées par un délimiteur et rangées dans des lignes (pour constituer des tuples). La plateforme accepte des fichiers compressés, avec certains formats de compression, ce qui contribue dans l'amélioration des performances de la plateforme. Afin d'étendre notre plateforme à d'autres formats de données, nous avons intégré une opération de conversion depuis ces formats sources vers le format pivot de Hadoop (csv ou txt). Notre plateforme accepte, dans son actuelle version, des données provenant de la base de données relationnelle MySQL, des données dans un modèle NoSQL de la base MongoDB et fichiers plats (csv et txt) ; d'autres formats peuvent être intégrés par la suite. En ce qui concerne le partitionnement, nous avons utilisé quelques méthodes. Une des méthodes implémentées fait partie de notre contribution, nous l'avons proposé pour assurer un certain équilibre dans la distribution de la charge des traitements sur le cluster.

Dans la phase T (Transforming), nous avons mis à la disposition des utilisateurs de la plateforme une série de transformateurs qui servent à reformater, nettoyer, vérifier, fusionner et consolider les données, avant qu'elles soient chargées. Cette phase a pour vocation de

rendre ces données homogènes et cohérentes avec la vue attendue par les utilisateurs de l'entrepôt de données.

Pour la dernière phase L (Load), la plateforme donne la liberté à l'utilisateur de choisir où et comment ses données résultantes seront finalement stockées. Il sera en mesure de choisir de sauvegarder ces résultats dans un fichier plat, compressé ou non, donnant vie à un cube de donnée, comme il peut les sauvegarder dans une base de données relationnelle MySQL ou MongoDB là où l'entrepôt de données peut résider.

Les trois étapes seront distribuées sur le cluster des servers, ce qui garantit un parallélisme complet tout le long du processus ETL. D'autre part, la plateforme offre une interface graphique simple et conviviale permettant aux utilisateurs de configurer un processus ETL et ce en termes de : sources des données, types d'extraction, technique de partitionnement, transformations à opérer sur les données, destination finale, etc.

2. Perspectives

Comme tout autre travail, le nôtre n'est pas parfait et mérite une continuité. Nous considérons qu'il faut étudier un certain nombre de perspectives qui rendent la plateforme plus complète. Nous citons ici les plus importants dont voici une synthèse :

- Etendre la plateforme par d'autres formats sources tels que les fichiers XML, Excel, d'autres bases de données telles que Oracle, Access et des bases de données NoSQL telles que HBase de Hadoop et Cassandra. L'idéal serait le développement d'extracteurs dédié chacun pour un format particulier afin d'éviter de recourir à la conversion vers le format pivot (fichier plat).
- Enrichir davantage la bibliothèque des transformateurs en donnant la main à l'utilisateur pour créer leurs fonctions spécifiques ;
- Intégrer un planificateur permettant d'exécuter un processus ETL périodiquement ; suite à un événement ou à une date donnée ;

D'autre part, pour connaitre les vraies limites et performances de la plateforme, nous avons besoin d'expérimentations sur une vraie infrastructure de type cluster en injectant de grandes quantités de données d'une dimension Big Data (TeraBytes, PetaBytes, HexaBytes).

Annexe A :
Hadoop

Annexe A : Hadoop

Annexe :

1. **Hadoop**

1.1 Définition

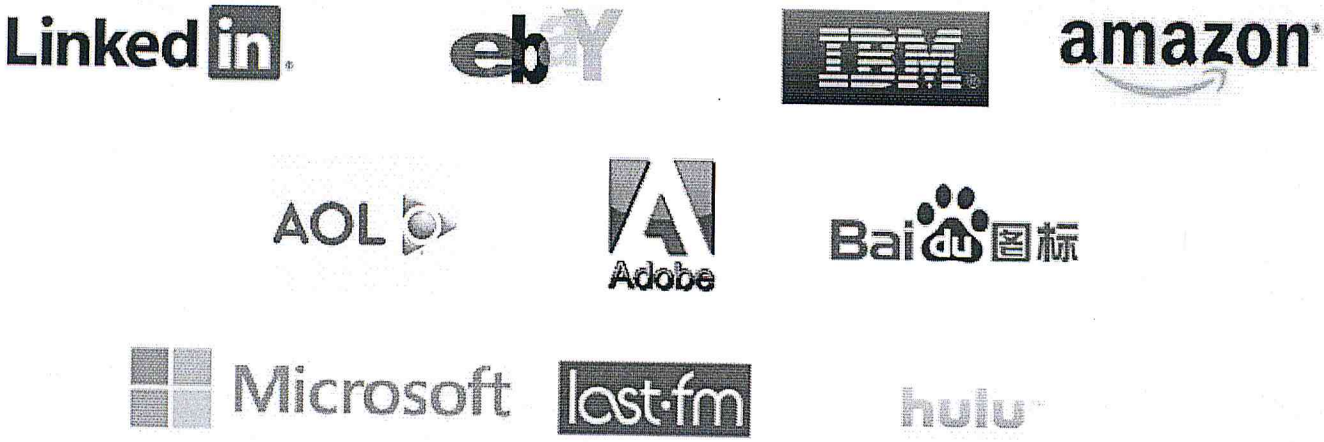
Hadoop est un framework open-source (de la fondation Apache) permet le traitement distribué de grands ensembles de données à travers des clusters d'ordinateurs utilisant un modèle de programmation simple. Il est conçu pour évoluer (scale up) à partir de serveurs simples à des milliers de machines, chacune d'eux offre un espace de stockage local. Plutôt que compter sur le matériel pour offrir une haute disponibilité, la bibliothèque Hadoop est conçue en soi pour détecter et traiter les défaillances de la couche d'application, donc fournir un service hautement disponible sur un cluster d'ordinateurs, chacun d'eux pouvant être sujettes à des défaillances. [W10]

Hadoop a été créé par Doug Cutting, le créateur de Lucene, a lancé le projet Hadoop, dans le cadre de la fondation Apache. Cutting serait ensuite embauché par Yahoo pour refondre toute la technologie de recherche du portail, en intégrant Hadoop et HDFS, qui reste intégralement open source tout en intégrant l'essentiel des développements financés par Yahoo.

Dans Hadoop, toutes les entités manipulées, les couples (clé,valeur), doivent être sérialisables, de sorte que l'on puisse les transmettre d'un serveur à un autre. Mais, puisqu'on a dit qu'il s'agissait de très gros volumes, il faut aussi optimiser l'utilisation de la bande passante sur le réseau. C'est pourquoi MapReduce est généralement utilisé en combinaison avec un système de gestion de fichiers distribué, dans le cas de Hadoop il s'agit de HDFS (détaillé dans la section 6.3 HDFS). Dans cette logique, chaque serveur est à la fois un outil de calcul et un outil de stockage. La fonction de Mapping cherchera alors à attribuer chaque tâche à un serveur qui stocke déjà les données correspondantes.

Hadoop le projet d'Apache s'étend sur deux sous projets :

- Hadoop MapReduce :
- Hadoop DFS :



http://wiki.apache.org/hadoop/PoweredBy

Prenons un aperçu sur le système Hadoop :

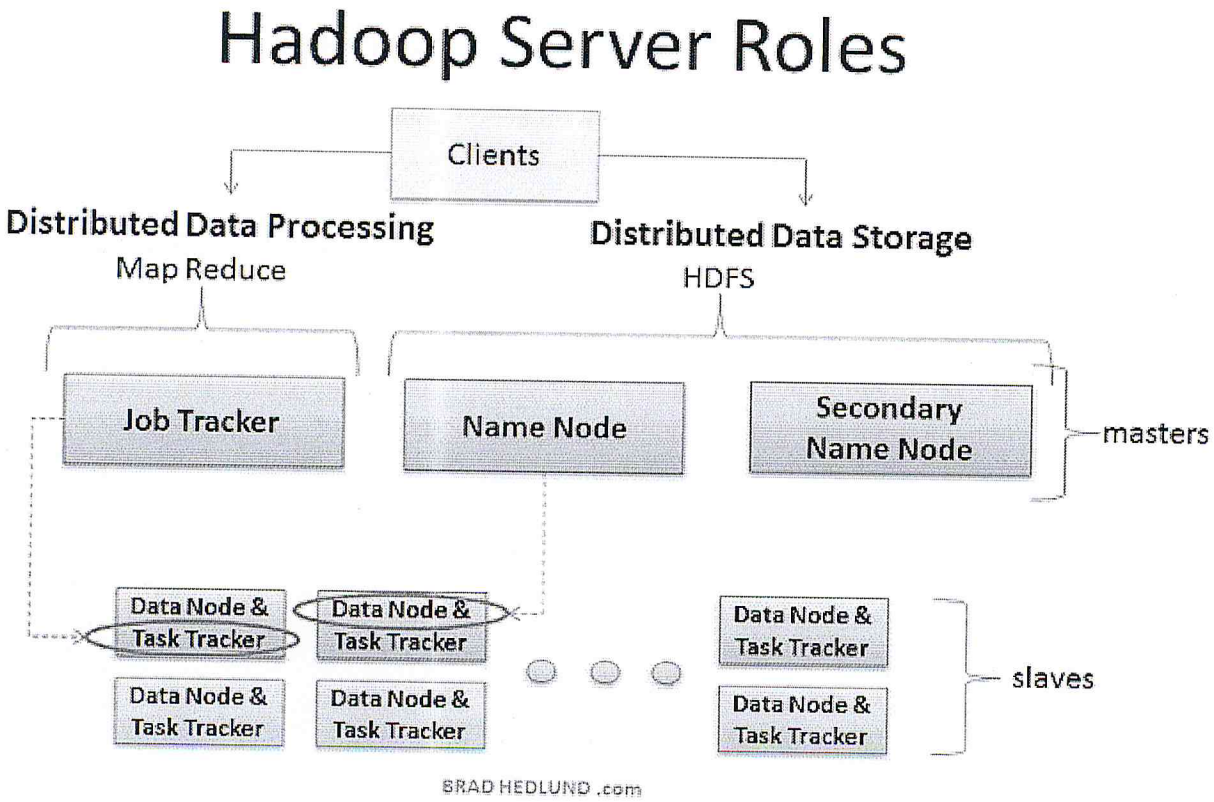


Figure 57 : Aperçu sur le système Hadoop

6.2 Moteur MapReduce de Hadoop

Le moteur MapReduce se compose d'un JobTracker, auquel les applications clientes soumettent des jobs MapReduce. Le JobTracker pousse le travail aux nœuds TaskTracker disponibles dans le cluster.

6.2.1 Job

Un job MapReduce est une unité de travail que le client veut être effectuée : elle consiste en : les données d'entrée, le programme MapReduce et les

informations de la configuration. Hadoop exécute le travail en le divisant en tâches, dont il existe deux types : tâches de Map et tâche de Reduce. [29]

6.2.2 JobTracker

Le JobTracker est le service de Hadoop qui affecte des tâches MapReduce aux nœuds spécifiques du cluster, idéalement les nœuds qui ont des données, ou au moins ceux qui sont dans le même rack.
-Les applications clientes soumettent des jobs au JobTracker.
-Le JobTracker parle au NameNode pour déterminer l'emplacement des données.
-Le JobTracker localise les nœuds TaskTracker qui sont près les données.
-Le JobTracker soumet le travail aux nœuds TaskTracker choisis.
-Les nœuds TaskTracker sont surveillés. S'ils ne présentent pas de signaux assez souvent, ils sont jugé avoir échoué et le travail est planifié sur un autre TaskTracker.
-Le TaskTracker notifiera le JobTracker lorsqu'une tâche échoue. Le JobTracker décide ce qu'il faut faire ensuite : il peut resoumettre le job ailleurs, ou il peut marquer cet enregistrement spécifique comme quelque chose à éviter, comme il peut même blacklister le TaskTracker comme non fiable.
-Lorsque le travail est terminé, le JobTracker met à jour son statut.
-Les applications clientes peuvent interroger le JobTracker pour plus d'informations.

Le JobTracker est un point de défaillance pour le service MapReduce Hadoop, c.-à-d. s'il tombe en panne, tous les jobs en cours sont arrêtés. Or, le JobTracker enregistre traces de ce qui se passe dans le système de fichiers. Quand il démarre, il recherche ces données, afin qu'il puisse reprendre le job là où il s'était arrêté. [W11]

6.2.3 TaskTracker

Le TaskTracker est un nœud du cluster qui accepte les tâches – Les opérations Map, Reduce et Shuffle – à partir d'un JobTracker. Chaque TaskTracker est configuré avec un ensemble de slots, ceux-ci indiquent le nombre de tâches qu'il peut accepter. Lorsque le JobTracker essaie de trouver un endroit pour planifier une tâche parmi les opérations MapReduce, il cherche d'abord un slot vide sur le même serveur qui héberge le DataNode contenant les données, et si non, il cherche un slot vide sur une machine dans le même rack.

Le TaskTracker génère un processus JVM séparé pour faire le travail, ça pour assurer que l'échec de processus n'arrête pas le TaskTracker. Le TaskTracker surveille ces processus générés, capturant la sortie et les codes de sortie. Une fois le processus est terminé, avec succès ou non, le tracker notifie le JobTracker. Les TaskTrackers envoient également des messages au JobTracker, généralement chaque quelques minutes, pour rassurer le JobTracker qu'il est encore en vie. Ces messages informent également le JobTracker du nombre de slots disponibles, de sorte que le JobTracker peut rester à jour avec où dans le cluster, le travail peut être délégué. [W11]

6.3 HDFS

6.3.1 Définition

Le Hadoop framework est muni d'un système de fichiers distribué HDFS extensible et portable écrit en Java. Il est conçu pour stocker de manière fiable de très gros fichiers sur plusieurs ordinateurs dans un grand cluster. Il est inspiré par le GoogleFileSystem GFS. Il 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.

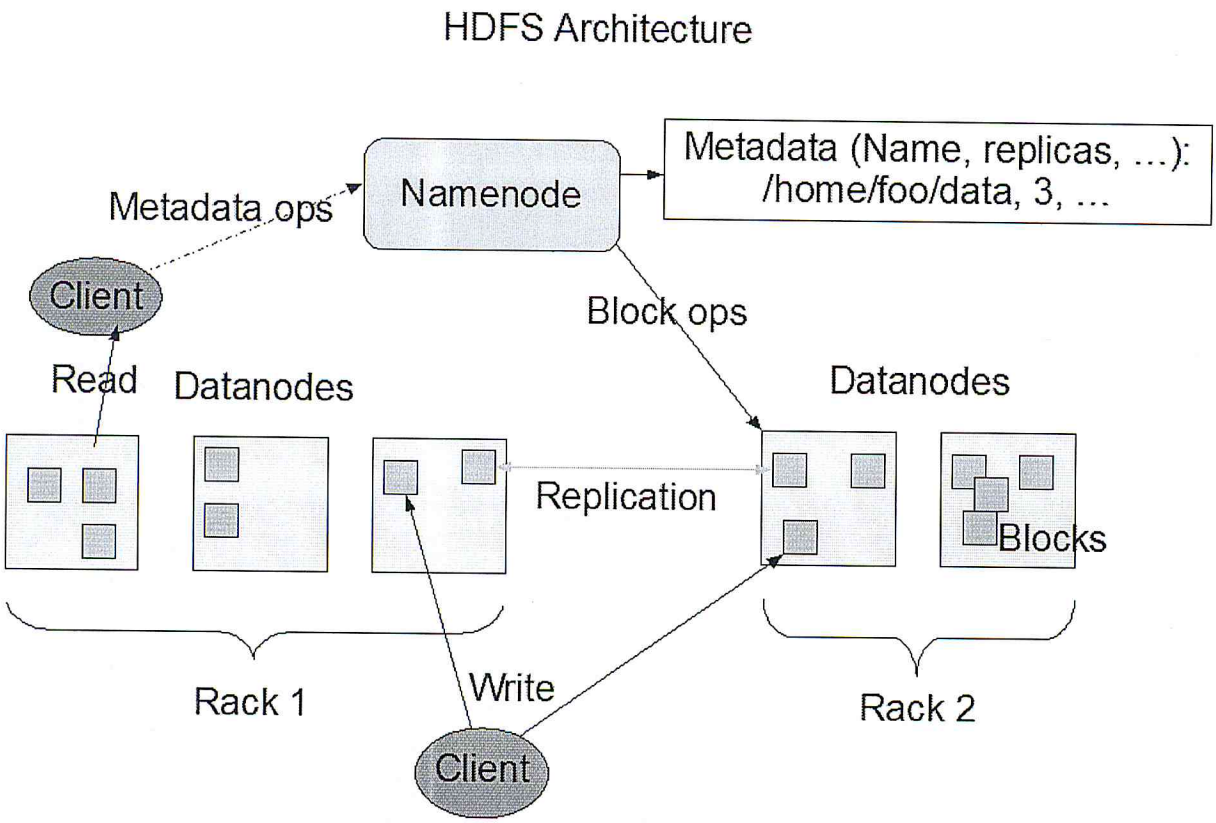


Figure 58 : Architecture de HDFS [W14]

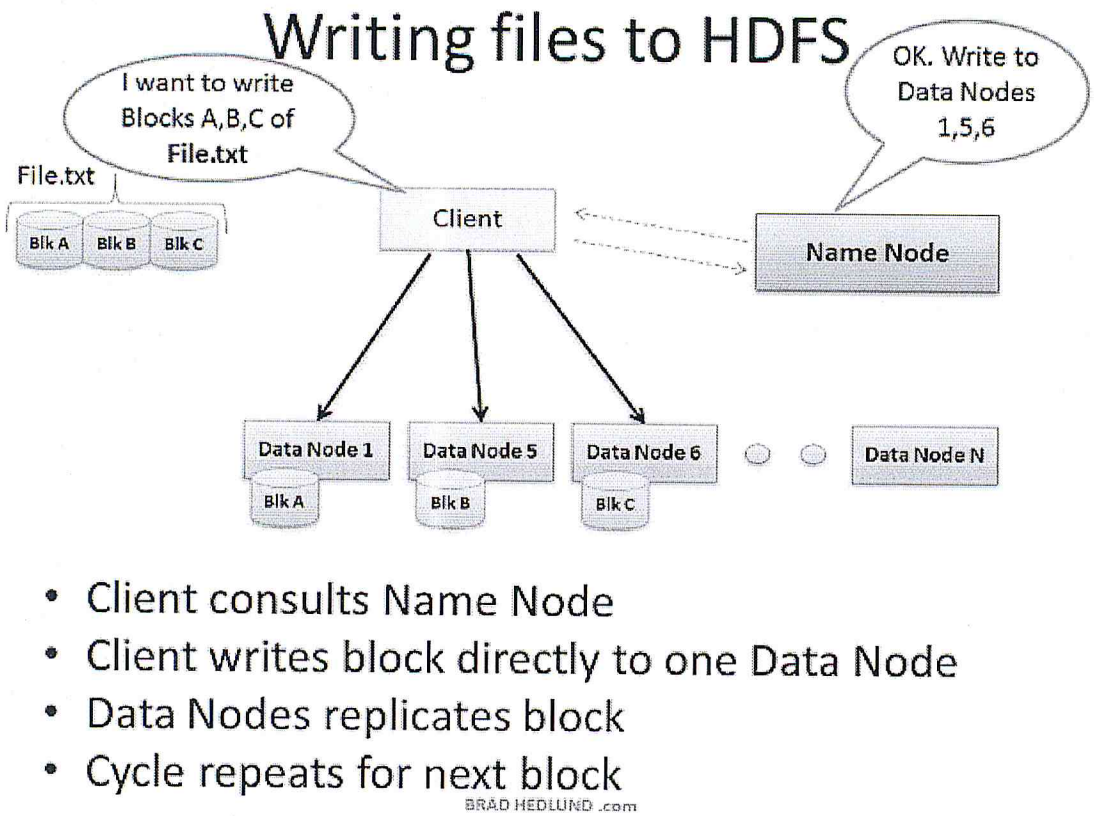


Figure 59 : Ecriture des données sur HDFS

6.3.2 Particularité de HDFS

HDFS présente de nombreuses similitudes avec les systèmes de fichiers distribués existants. Cependant, ces différences sont importantes : HDFS est très tolérant aux pannes et est conçu pour être déployé sur un matériel à faible coût. HDFS fournit un accès haut débit aux données de l'application et est adapté pour les applications qui ont des ensembles de données volumineux. HDFS détend quelques exigences POSIX pour permettre un accès continu aux données du système. HDFS a été initialement construit comme infrastructure pour le projet de moteur de recherche Apache Nutch web. HDFS est maintenant un Apache Hadoop sous-projet.

6.3.3 Distribution des données

Dans un cluster Hadoop, les données sont distribuées à tous les nœuds du cluster lors leur chargement dans le système. Le Système des fichiers distribué (Distributed File System HDFS) va diviser les fichiers de données volumineux en morceaux qui sont gérés par les différents nœuds du cluster. En plus de cela, chaque morceau est répliqué sur plusieurs machines, de sorte qu'une défaillance de la machine seule ne conduit pas à toutes les données ne seraient pas disponibles. Un système de surveillance active puis re-réplique les données en réponse aux défaillances du système qui peuvent résulter en stockage partiel. Même si les morceaux de fichiers sont répliqués et répartis sur plusieurs machines, ils forment un seul espace de noms, de sorte que leurs contenus soient accessibles à tous.

6.3.4 Localité des données

Les données sont conceptuellement orientées-enregistrement (record-oriented) dans le framework Hadoop. Les fichiers d'entrée individuelles sont divisés en lignes (ou en d'autres formats spécifiques à la logique d'application). Chaque processus s'exécutant sur un nœud du cluster traite alors un sous-ensemble de ces enregistrements. Le framework Hadoop planifie ensuite ces processus à proximité de l'emplacement des données (enregistrements) en utilisant les connaissances du système de fichiers HDFS. Comme les fichiers sont répartis à travers le système de fichiers distribué comme des morceaux, chaque processus en cours d'exécution sur un nœud fonctionne sur un sous-ensemble des données. Les données exploitées par un nœud sont choisies en fonction de leur localité à ce nœud (les plus proches): la plupart des données sont lues à partir du disque local directement dans le CPU afin de réduire la pression sur la bande passante du réseau et prévenir les transferts réseau inutiles. Cette stratégie de *déplacement les calculs aux données*, au lieu de *déplacer les données aux calculs* permet au Hadoop de réaliser une localité de données élevée qui à son tour, résulte dans une haute performance [W26].

6.3.5 Composants du HDFS

NameNode

HDFS a une architecture maître/esclave. Un cluster HDFS se compose d'une seule NameNode, il gère l'espace de noms (namespace) du système de fichiers et réglemente l'accès aux fichiers par les clients. Le NameNode est la pièce centrale d'un système de fichiers HDFS. Il conserve l'arborescence des répertoires de tous les fichiers du système de fichiers, et suit où dans le cluster les données du fichier sont conservées. Il ne stocke pas lui-même les données de ces fichiers.

Le NameNode est un point unique de défaillance pour le Cluster HDFS. HDFS n'est pas un système de haute disponibilité. Lorsque le NameNode s'arrête le système de fichiers se met hors ligne.

Les applications clientes parlent à la NameNode quand ils le souhaitent pour localiser un fichier, ou quand ils veulent ajouter/copier/déplacer/supprimer un fichier. Le NameNode répond aux requêtes réussisses en retournant une liste de serveurs pertinents appelés servers de DataNode où vivent les données.

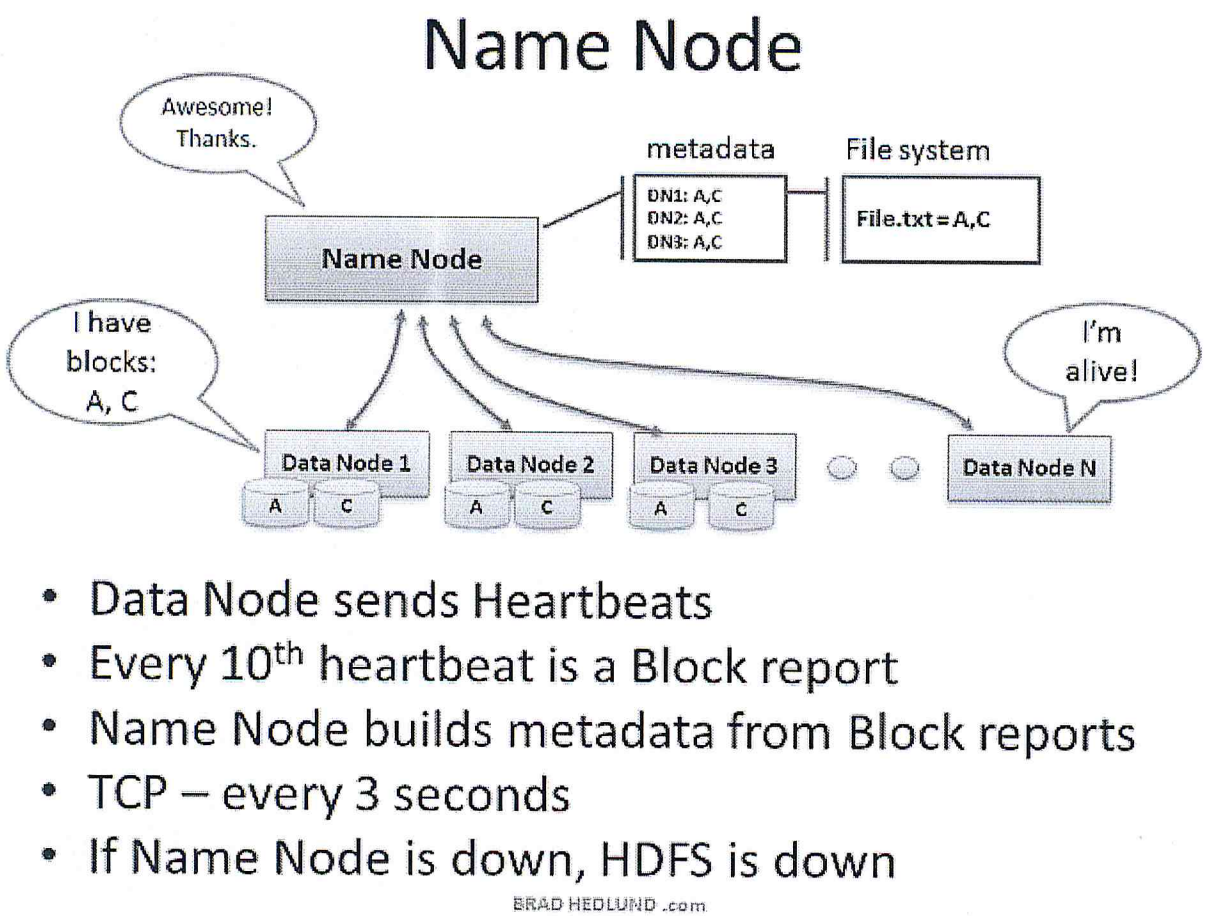


Figure 60 : Name Node

DataNode

Un système de fichiers fonctionnel possède plus d'un DataNode, avec les données répliquées à travers eux.

Au démarrage, un DataNode connecte au NameNode filant jusqu'à ce que le service soit en place. Il répond ensuite aux demandes de la NameNode pour les opérations du système de fichiers.
Les applications clientes peuvent parler directement à un DataNode, une fois le NameNode a fourni l'emplacement des données. De même, les opérations MapReduce affermées à des instances de TaskTracker près d'une DataNode, parlent directement à la DataNode pour accéder aux fichiers. Les instances de TaskTracker peuvent, en effet doivent, être déployées sur les mêmes serveurs qui accueillent les instances de DataNode, afin que les opérations MapReduce soient effectuées à proximité des données.

Les instances de DataNode peuvent parler les unes aux autres, et c'est ce qu'ils font quand ils sont répliquent des données.

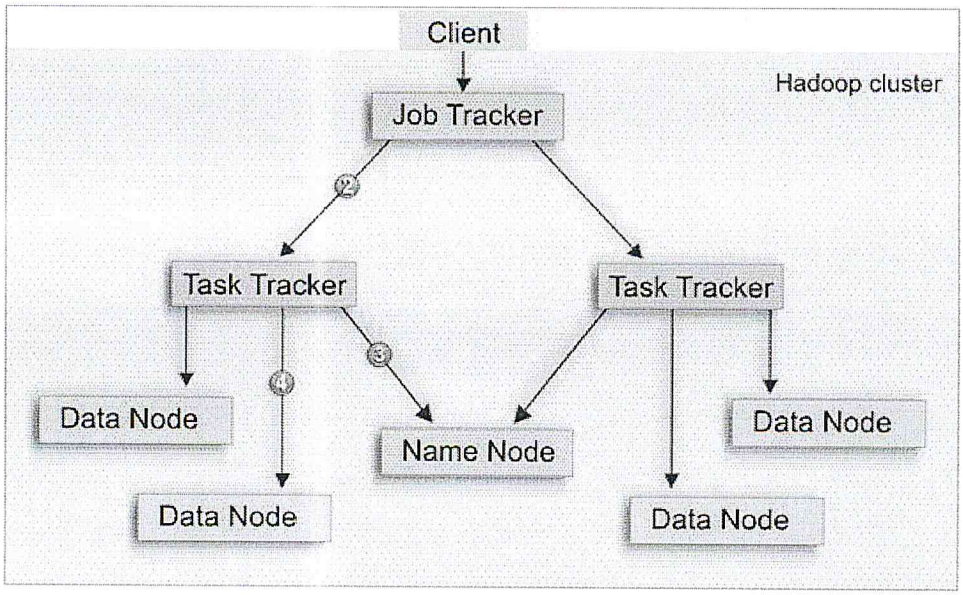


Figure 61 : Présentation générale des éléments de Hadoop [W15]

6.4 Flux de données

6.4.1 Les divisions Splits

Il existe deux types de nœuds qui contrôlent le processus d'exécution de la tâche : un JobTracker et un certain nombre de TaskTrackers. Le JobTracker coordonne tous les jobs exécutés sur le système en planifiant des tâches à exécuter sur TaskTrackers. Tasktrackers exécutent des tâches et envoient des rapports de la progression au JobTracker, qui tient un registre de l'état d'avancement de chaque job. Si une tâche échoue, le JobTracker peut la re-planifier sur un autre TaskTracker.

Hadoop divise les données d'entrée d'un job MapReduce en morceaux de taille fixe appelés input splits, ou simplement splits. Hadoop crée une tâche de Map pour chaque split, qui exécute la fonction de Map définie par l'utilisateur pour chaque enregistrement (record). Avoir plusieurs splits signifie le temps nécessaire pour traiter chaque split est court par rapport au temps de traitement de toute l'entrée. Donc, si nous traitons les splits de petites taille (on parle de big data, une petites taille n'est pas évidement seulement quelque mégaoctets, d'ailleurs les trop petites tailles influent négativement sur le process en Hadoop) en parallèle, le traitement est meilleur en termes d'équilibrage de charge (load-balance), puisque une machine plus rapide sera en mesure pour traiter proportionnellement plus de splits au cours du job qu'une machine plus lente.

Même si les machines sont identiques, les processus défaillants ou d'autres jobs qui s'exécutent simultanément rend l'équilibrage de charge souhaitable, et la qualité de l'équilibrage de charge augmente comme les splits deviennent à granularité plus fin (fine-grained).

D'autre part, si les splits sont trop petits, puis les frais généraux de la gestion des splits et de la création des tâches Map commencent à dominer le temps total d'exécution du job. Pour la plupart des jobs, une bonne taille de split tend à être la taille d'un bloc HDFS, 64 Mo par

défaut, mais cette taille puisse être modifiée pour le cluster (pour tous les fichiers nouvellement créés), ou spécifiée lors de la création de chaque fichier. Hadoop fait de son mieux pour exécuter la tâche Map sur un nœud où les données d'entrée résident. C'est ce qu'on appelle l'optimisation de la localité des données. Il devrait maintenant être clair pourquoi la taille optimale du split est la même que la taille du bloc : c'est la plus grande taille d'entrée qui peut être garantie d'être stockée sur un seul nœud. Si le split s'étend sur deux blocs, une partie de split faudrait être transférée via le réseau au nœud exécutant la tâche Map, ce qui est nettement moins efficace que l'exécution de toute la tâche en utilisant des données locales. [29]

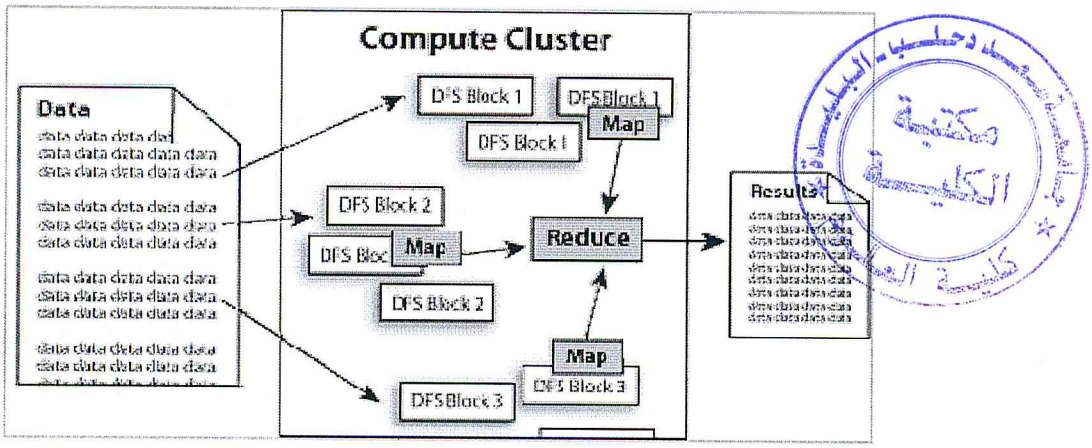


Figure 62 : Distribution des blocks de données dans le cluster [W13]

6.4.2 La sortie des Maps

Les tâches de Map écrivent leurs sorties sur le disque local, et non pas sur l'HDFS. Pourquoi est-ce? La sortie de Map est une sortie intermédiaire: elle est traitée par les tâches de Reduce pour produire le résultat final, et une fois que le job est terminé, la sortie de Map peut être jetée. Donc le stockant dans HDFS, avec la réplication serait exagéré. Si le nœud exécutant la tâche Map échoue avant que la sortie du Map soit consommée par la tâche Reduce, alors Hadoop va automatiquement relancer la tâche Map sur un autre nœud pour recréer la sortie du Map. [29]

6.4.3 La sortie des Reduces

Les tâches Reduce n'ont pas l'avantage de la localité des données, l'entrée à un seul tâche Reduce est normalement la sortie de tous les Mappers. Par conséquent, les sorties des Maps triées doivent être transférées à travers le réseau vers le nœud où la tâche Reduce est en cours d'exécution, où elles sont fusionnées, puis transmises à la fonction Reduce définie par l'utilisateur. La sortie du Reduce est normalement stockée dans HDFS pour la fiabilité. [29]

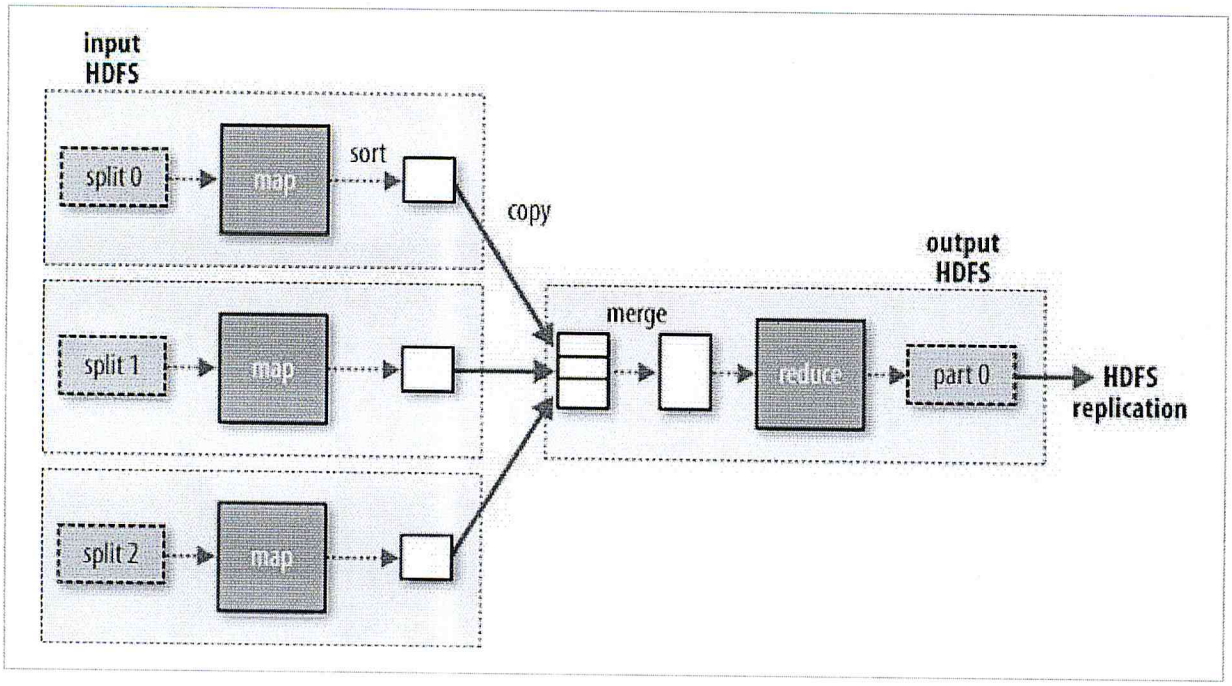


Figure 63 : Flux de données MapReduce avec un seul Reducer [29]

6.4.4 Partitionnement

Dans le cas de plusieurs Reducers, le Partitionnement est le processus de détermination quelle instance du Reducer recevra les clés et les valeurs intermédiaires. Chaque Mapper doit déterminer pour toute sa sortie, paires (clé,valeur), quel Reducer va la recevoir. Il est nécessaire que pour n'importe quelle clé, quelle que soit l'instance du Mapper qu'elle l'a générée, la partition de destination soit la même. Si la clé "Algérie" est générée dans deux paires distinctes (clé,valeur), ces dernières doivent être réduites ensemble. Il est également important, pour des raisons de performance, que les Mappers puissent partitionner les données indépendamment ; ils ne doivent jamais avoir besoin d'échanger des informations entre eux pour déterminer la partition pour une clé particulière.

Hadoop utilise une fonction appelée Partitioner pour déterminer à quelle partition une paire (clé,valeur) sera soumise. Une partition fait référence à toutes les paires (clé,valeur) qui seront envoyées à une seule tâche Reduce. Hadoop MapReduce détermine le moment où le job commence et sur combien de partitions il va diviser les données. Si une vingtaine de tâches Reduce doivent être exécutées, puis vingt partitions doivent être remplies.

Le partitionnement peut être contrôlé par l'utilisateur, mais normalement le Partitioner par défaut, utilisant une fonction de *hachage* qui fonctionne très bien. [W12]

Le flux de données pour le cas général de multiple Reducers est illustré dans la figure ci-dessous :

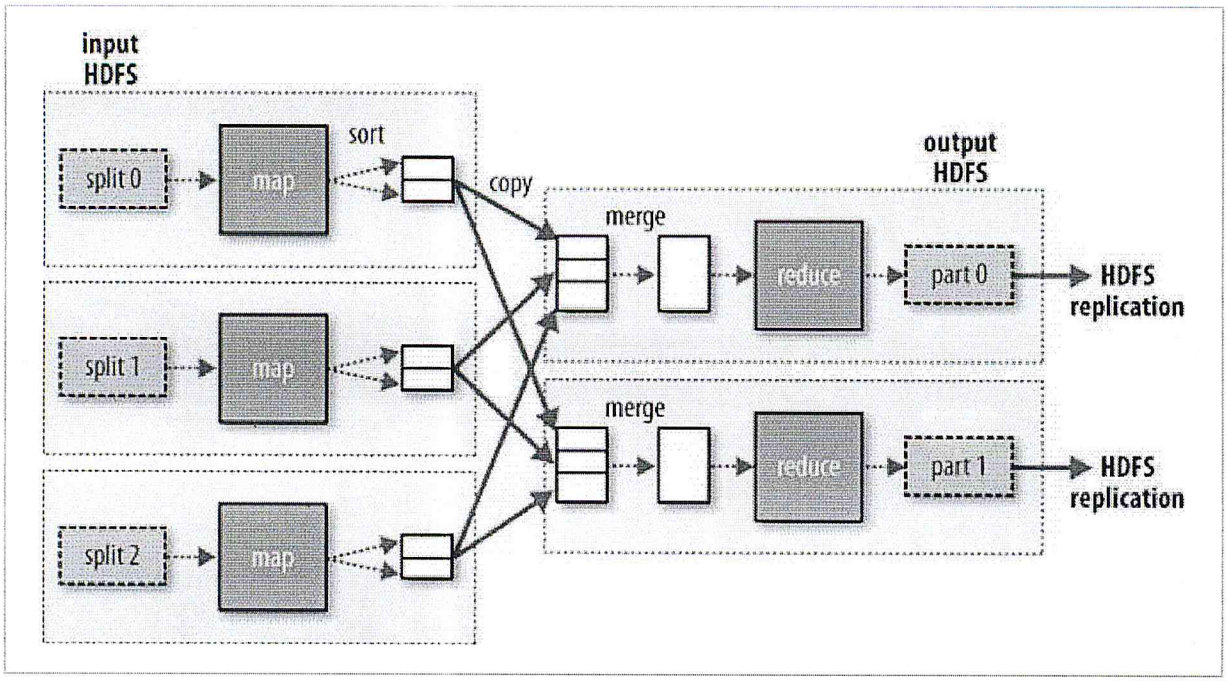


Figure 64 : Flux de données MapReduce avec plusieurs Reducers [29]

Finalement, il est possible d'avoir zéro tâche Reducer. Cela peut être valable lorsque vous n'avez pas besoin du Shuffle puisque le traitement peut être réalisé entièrement en parallèle sans avoir besoin des agrégations. Dans ce cas, le seul transfert de données hors-nœud, c'est quand les tâches de Map écrivent sur l'HDFS. [29]

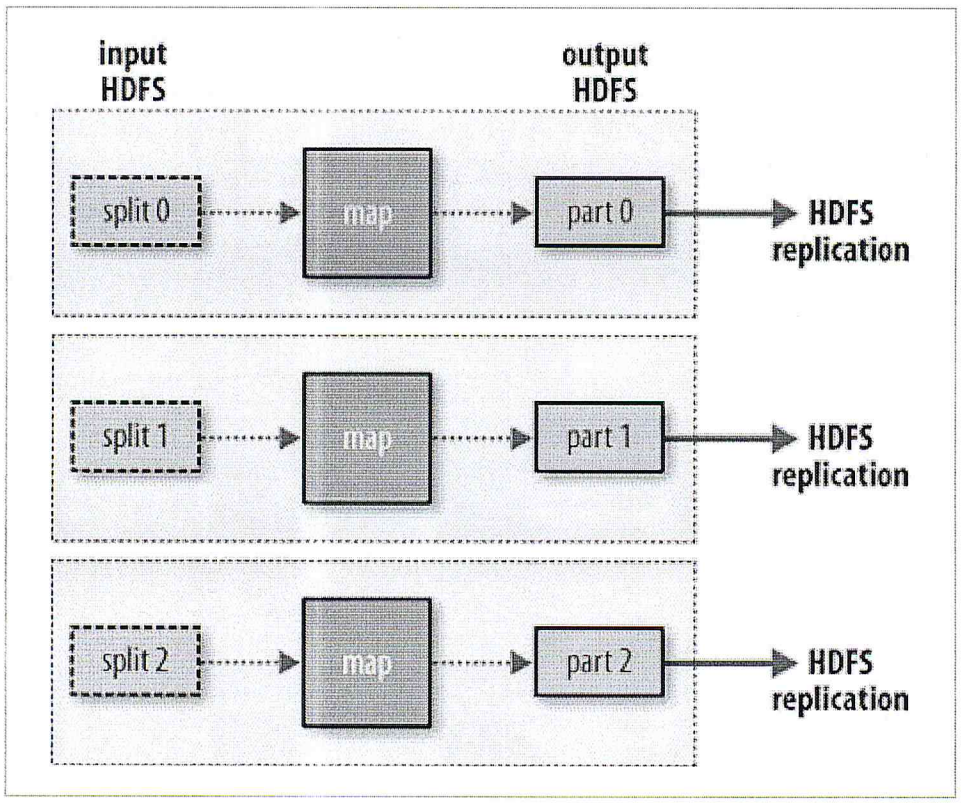


Figure 65 : Flux de données MapReduce avec aucun Reducer [29]

Annexe B :
Installation de Hadoop sur Ubuntu

Annexe B : Installation de Hadoop sur Ubuntu

Nous allons expliquer comment installer le framework Hadoop sur une machine Ubuntu Linux [W14]

1. Prérequis

Cette installation a été testée avec les versions logicielles suivantes :

- Ubuntu 12.04 /12.10

- Hadoop1.0.4 /1.1.2

2 Sun Java 6

Hadoop nécessite l'installation de Java 1.5 ou plus (aka Java 5). Nous avons utilisé Java 1.6 (aka Java 6) pour l'exécution de Hadoop. Donc nous allons décrire l'installation de Java 1.6 :

```
# Nettoyez le historique de openjdk
$ sudo apt-get purge openjdk

# Ajouter un nouveau référentiel
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:ferramroberto/java

# Mettre à jour la liste des sources
$ sudo apt-get update

# Installation de Sun JDK Java6
$ sudo apt-get install sun-java6-jdk

# Sélectionnez Java de Sun par défaut sur votre machine
 $ sudo update-java-alternatives –s java-6-sun
```

Après l'installation, une vérification rapide si le JDK de Sun est correctement mis en place :

```
user@ubuntu:~# java –version
java version « 1.6.0_20 »
Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
Java HotSpot(TM) Client VM (build 16.3-b01, mixed mode, sharing)
```

3 Utilisateur du système Hadoop

Il est recommandé d'utiliser un compte d'utilisateur dédié à Hadoop pour l'exécution de Hadoop car il permet de séparer l'installation de Hadoop des autres applications logicielles et des comptes d'utilisateurs fonctionnant sur la même machine (penser : la sécurité, les permissions, les sauvegardes, etc.)

```
1  $ sudo addgroup 131adoop
2  $ sudo adduser –ingroup 131adoop hduser
```

Cela va ajouter l'utilisateur hduser et le groupe Hadoop sur votre machine locale.

4 Configuration de SSH

Hadoop nécessite un accès SSH pour gérer ses nœuds, c'est à dire des machines distantes, donc nous avons besoin de configurer l'accès SSH à localhost pour l'utilisateur hduser que nous avons créé dans la section précédente.

1. Tout d'abord, nous devons générer une clé SSH pour l'utilisateur hduser :

```
1  user@ubuntu:~$ su – hduser
2  hduser@ubuntu:~$ ssh-keygen –t rsa –P «  »
3  Generating public/private rsa key pair.
4  Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
5  Created directory '/home/hduser/.ssh'.
6  Your identification has been saved in /home/hduser/.ssh/id_rsa.
7  Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
8  The key fingerprint is:
9  9b:82:ea:58:b4:e0:35:d7:ff:19:66:a6:ef:ae:0e:d2 hduser@ubuntu
10 The key's randomart image is:
11 […snipp…]
```

2. Vous devez activer l'accès SSH sur votre machine locale avec la nouvelle clé créé :

```
1  hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

3. La dernière étape consiste à tester la configuration SSH en vous connectant à votre ordinateur local avec l'utilisateur hduser :

```
1  hduser@ubuntu:~$ ssh localhost
2  The authenticity of host 'localhost (::1)' can't be established.
3  RSA key fingerprint is d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
4  Are you sure you want to continue connecting (yes/no)? yes
5  Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
6  Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686
7  GNU/Linux
   Ubuntu 12.04
```

5 Désactivation de Ipv6

1. Pour désactiver Ipv6 sur Ubuntu 10.04 LTS, ouvrir 4 "/etc/sysctl.conf " dans l'éditeur de votre choix et ajouter les lignes suivantes à la fin du fichier :

/etc/sysctl.conf

```
1  # disable ipv6
2  net.ipv6.conf.all.disable_ipv6 = 1
3  net.ipv6.conf.default.disable_ipv6 = 1
4  net.ipv6.conf.lo.disable_ipv6 = 1
```

2. Redémarrer le PC.

3. vérifier si Ipv6 est activé sur votre machine avec la commande suivante :

```
1  $ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

6 Configuration et installation de Hadoop

6.1 Installation

Télécharger Hadoop et d'extraire le contenu du package Hadoop à un endroit de votre choix. On a choisi "/usr / local / Hadoop". Assurez-vous de changer le propriétaire de tous les fichiers de l'utilisateur hduser et de groupe Hadoop :

```
1  $ cd /usr/local
2  $ sudo tar xzf 132adoop-1.0.3.tar.gz
3  $ sudo mv 132adoop-1.0.3 hadoop
4  $ sudo chown –R hduser:132adoop hadoop
```

6.2 Mise à jour $HOME/.bashrc

Ajoutez les lignes suivantes à la fin du fichier $HOME /.bashrc del'utilsateur hduser :

$HOME/.bashrc

```
1  # Set Hadoop-related environment variables
2  export HADOOP_HOME=/usr/local/133adoop
3
4  # Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
5  export JAVA_HOME=/usr/lib/jvm/java-6-sun
6
7  # Some convenient aliases and functions for running Hadoop-related commands
8  unalias fs &> /dev/null
9  alias fs="133adoop fs"
10 unalias hls &> /dev/null
11 alias hls="fs –ls"
12 # If you have LZO compression enabled in your Hadoop cluster and
13 # compress job outputs with LZOP (not covered in this tutorial):
14 # Conveniently inspect an LZOP compressed file from the command
15 # line; run via:
16 #
17 # $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
18 #
19 # Requires installed 'lzop' command.
20 #
21 lzohead () {
22     133adoop fs –cat $1 | lzop –dc | head -1000 | less
23 }
24
```

```
25 # Add Hadoop bin/ directory to PATH
26 export PATH=$PATH:$HADOOP_HOME/bin
```

6.3 Configuration de HDFS

6.3.1 Mise à jours de fichier hadoop-env.sh

Ouvert "conf/Hadoop-env.sh" dans l'éditeur de votre choix (si vous avez utilisé le chemin d'installation dans ce annexe, le chemin complet est "/usr/local/Hadoop/conf/Hadoop-env.sh") et définissez la variable d'environnement JAVA_HOME pour le répertoire de JDK/JRE Sun 6.

Changer les lignes suivantes :

conf/Hadoop-env.sh

```
1 # The java implementation to use.  Required.
2 # export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

Par:

conf/Hadoop-env.sh

```
1 # The java implementation to use.  Required.
2 Export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

6.3.2 Création de dossier temporaire pour Hadoop

Nous allons créer le répertoire " app/hadoop/tmp" où Hadoop va stocker ses fichiers de données :

```
1 $ sudo mkdir –p /app/133adoop/tmp
2 $ sudo chown hduser:133adoop /app/133adoop/tmp
3 # …and if you want to tighten up security, chmod from 755 to 750…
4 $ sudo chmod 750 /app/133adoop/tmp
```

6.3.3 Mise à jours des fichiers conf/*-site.xml

1. conf/core-site.xml :

conf/core-site.xml

```
1 <property>
2   <name>134adoop.tmp.dir</name>
3   <value>/app/134adoop/tmp</value>
4   <description>A base for other temporary directories.</description>
5 </property>
6
7 <property>
```

```
8   <name>fs.default.name</name>
9   <value>hdfs://localhost:54310</value>
10  <description>The name of the default file system.  A URI whose
11  scheme and authority determine the FileSystem implementation.  The
12  uri's scheme determines the config property (fs.SCHEME.impl) naming
13  the FileSystem implementation class.  The uri's authority is used to
14  determine the host, port, etc. for a filesystem.</description>
15 </property>
```

2. conf/mapred-site.xml :

conf/mapred-site.xml

```
1  <property>
2   <name>mapred.job.tracker</name>
3   <value>localhost:54311</value>
4   <description>The host and port that the MapReduce job tracker runs
5   at.  If "local", then jobs are run in-process as a single map
6   and Reduce task.
7   </description>
8  </property>
```

3. conf/hdfs-site.xml :

conf/hdfs-site.xml

```
1  <property>
2   <name>dfs.replication</name>
3   <value>1</value>
4   <description>Default block replication.
5   The actual number of replications can be specified when the file is created.
6   The default is used if replication is not specified in create time.
7   </description>
8  </property>
```

6.3.4 Formatage du système de fichiers HDFS via le NameNode

La première étape de la mise en service de votre installation Hadoop est de formater le système de fichiers Hadoop.

Notes : ne pas formater un système de fichiers Hadoop à fonctionner car vous allez perdre toutes les données actuellement dans le cluster (en HDFS) !

Pour formater le système de fichiers (qui initialise simplement le répertoire spécifié par la variable dfs.name.dir), exécutez la commande :

```
1  hduser@ubuntu:~$ /usr/local/134adoop/bin/134adoop namenode –format
```

Annexe B : Installation de Hadoop sur Ubuntu

7 Vérification de *fonctionnement*

7.1 Démarrage des services de Hadoop

1. Pour démarrer le JobTracker, le Namenode, le Datanode, et les TaskTrackers sur votre machine exécutée la commande :

```
1 | hduser@ubuntu:~$ /usr/local/135adoop/bin/start-all.sh
```

2. vérifier avec netstat si Hadoop est à l'écoute sur les ports configurés :

```
1 | hduser@ubuntu:~$ sudo netstat –plten | grep java
```

7.2 Arrêt des services de Hadoop

Pour arrêter tous les serveurs fonctionnant sur votre machine exécutée la commande :

```
1 | hduser@ubuntu:~$ /usr/local/135adoop/bin/stop-all.sh
```

8 Exécution d'un job MapReduce

Nous allons maintenant lancer votre premier emploi Hadoop MapReduce. Nous allons utiliser l'exemple de WordCount qui lit les fichiers texte et compte combien de fois les mots se produisent. L'entrée est des fichiers texte et la sortie est fichiers texte, dont chaque ligne contient un mot et le compte de combien de fois se produisent. Plus d'informations sur ce qui se passe dans les coulisses est disponible sur le Wiki Hadoop.

1. stocker les fichiers dans un répertoire temporaire local de son choix, par exemple "/tmp /wordcount"
2. Redémarrez votre cluster Hadoop si ce n'est pas déjà en cours d'exécution.

```
1 | hduser@ubuntu:~$ /usr/local/135adoop/bin/start-all.sh
```

3. Avant d'exécuter le travail Réel MapReduce, nous devons d'abord copier les fichiers à partir de notre système de fichiers local à HDFS d'Hadoop :

```
1 | hduser@ubuntu:/usr/local/hadoop/bin/hadoop dfs -copyFromLocal /tmp/wordcount
  | /user/hduser/wordcount
```

4. Maintenant, on peut exécuter le travail de l'exemple WordCount :

```
1 | hduser@ubuntu:/usr/local/hadoop/bin/Hadoop jar 135adoop-examples-1.0.4.jar
  | wordcount /user/hduser/wordcount /user/hduser/wordcount-output
```

Cette commande va lire tous les fichiers dans le répertoire HDFS "/user/hduser/wordcount", traiter et stocker le résultat dans le répertoire HDFS "/user/hduser/wordcount-output".

5. Récupérer la résultat du travail a partir de HDFS avec cette commande :

```
1  hduser@ubuntu:/usr/local/hadoop/bin/hadoop dfs -cat /user/hduser/wordcount-output/part-r-00000
```

Pour lire le fichier à partir de fichier local il faut le copier les fichier de HDFS dans le système de fichiers local par cette commande :

```
1  hduser@ubuntu:/usr/local/hadoop$ mkdir /tmp/wordcount-output
2  hduser@ubuntu:/usr/local/hadoop/bin/hadoop dfs -getmerge /user/hduser/wordcount-output /tmp/wordcount-output
```

La commande fs-getmerge sera tout simplement concaténer tous les fichiers qu'il se trouve dans le répertoire que vous spécifiez.

9 Interfaces web de Hadoop

Hadoop est livré avec plusieurs interfaces web qui sont par défaut dans le fichier "conf/ Hadoop-default.xml" disponibles aux endroits suivants :

1. http ://localhost:50070/ – interface web pour les HDFS Namenodes
2. http ://localhost:50060/ – interface web pour les tasktracker.
3. http ://localhost:50030/ - interface web pour le travail MapReduce trackers.

Ces interfaces Web fournissent des renseignements concis sur ce qui se passe dans votre cluster Hadoop.

Annexe C : Solutions basées sur Hadoop

Annexe C : Solutions basés sur Hadoop

1. Hive :

Apache Hive est un système d'entrepôt de données pour Hadoop qui facilite la synthèse des données, les requêtes ad-hoc et l'analyse de grands ensembles de données stockées dans les systèmes de fichiers compatibles Hadoop. Hive fournit un mécanisme pour la structure du projet sur ces données et pour l'interrogation des données en utilisant un langage de type SQL appelé HiveQL. Dans le même temps, ce langage permet également aux programmeurs de MapReduce traditionnel de brancher leurs Mappers et Reducers personnalisés lorsqu'il n'est pas pratique ou inefficace pour exprimer une certaine logique en HiveQL. [W10]

2. Pig :

Apache Pig est une plateforme d'analyse de grands ensembles de données qui se compose d'un langage de haut niveau pour exprimer des programmes d'analyse de données, couplé avec une infrastructure pour l'évaluation de ces programmes. La propriété saillants des programmes de Pig est que leur structure se prête à un parallélisation substantielle, qui à son tour leur permet de manipuler des ensembles de données très volumineux. À l'heure actuelle, la couche d'infrastructure de Pig se compose d'un compilateur qui produit des séquences des programmes MapReduce pour lesquels il existe déjà des implémentations parallèles à grande échelle (par exemple, le sous-projet Hadoop). La couche de langage de Pig se compose actuellement d'un langage textuel appelé Pig Latin. [W10]

3. Hbase :

Apache Hbase est la base de données de Hadoop ; c'est un grand magasin de données évolutif et distribué. Vous l'utilisez lorsque vous avez besoin d'une lecture/écriture aléatoire et en temps réel de vos Big Data. L'objectif de ce projet est l'hébergement de très grandes tables – des milliards de lignes x millions de colonnes- placées sur des clusters de matériel standard. Apache Hbase est un magasin open-source, distribué, versionné et orienté colonne modelé sur le Bigtable de Google : Un système de stockage distribué pour les données structurées. Juste comme Bigtable qui s'appuie sur le stockage de données distribuées fourni par le GFS (Google File System), Apache Hbase offre des capacités comme le Bigtable au-dessus de Hadoop et HDFS. [W10]

Bibliographie

Bibliographie

1. Références bibliographiques

[01] Gorry, G.A. Scott Morton, M. Sloan « Management Review », 1971, Vol. 12, no1, pp.55-70.

[02] K. Smaïli, « Systèmes d'information décisionnels et Datamining », 2008.

[03] Bill Inmon, « Building the Data Warehouse »,1996.

[04] Jambu, M., « Introduction au Data Mining. Analyse intelligente des données », Eyrolles, Paris, 1999, 144p.

[05] Kimball R., Ross M., « The Data Warehouse Toolkit », Wiley, New York, deuxième édition, 2002.

[06] Louisa Demmou, « Exploration de problèmes de performance d'un entrepôt de données », 2010.

[07] Ravat et al, 2000 Ravat F., Teste O., « An Object Data Warehousing Approach: a Web Site Repository», Dans 4th East-European Conference on Advances in Databases and tel-00549421, version 1 - 21 Dec 2010.

[08] Codd E. F., Codd S.B., Salley C.T., « Providing OLAP (On Line Analytical Processing) to Users-Analysts: An IT Mondate", Rapport technique, E.F. Codd and Associates, 1993.

[09] Choong Y. W., Laurent D. and Marcel P. « Computing appropriate representations for multidimensional data ». Data & Knowledge Engineering, Vol. 45, N. 2, p. 181-203, mai 2003.

[10] GHOZZI J. F, ZURFLUH.G, « Conceptions et manipulation de bases de données dimensionnelles à contraintes », 2004.

[11] Paulraj Ponniah, « Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals », John Wiley ET Sons 2001, Inc.

[12] Villacampa F. « Olap: analyser les données de l'entreprise », Décision Micro, 05/08/2002.

[13] Christophe Jermann – Florin FRANCHETEAU, « Rapport de stage, Etdude des ETL open Source », 2007/2008.

[14] Robert J Davenport, « Data Academy, ETL vs ELT, part of the serie of the in source Commercial Aspects of Bi discussion papers », June 2008.

[15] J-F. Desnos, « Entrepôt de données Introduction ».

[16] Richard Dobbs, Seoul James, Charles Roxburgh, Susan Lund « Big data: The next frontier for innovation, competition and productivity. Technical » June 2011.

Bibliographie

[17] Stefane Fermigier, « BIG DATA & OPEN SOURCE: UNE CONVERGENCE INÉVITABLE ».

[18] Oracle, « Big data for the Enterprise White Paper », 2012.

[19] Redhat, « the Five Must-haves of Big data storage ».

[20] DATASTAX corporation, « Big Data: Beyond the Hype, white paper », march 2012.

[21] Jimmy Lin, Chris Dyer, « Data-Intensive Text Processing with MapReduce », Manuscript prepared April 11, 2010 University of Maryland, College Park.

[22] Patrice BERTRAND – Smile, « plateformes web Hautes performance Principes d'architecture et outils open-source ».

[23] Dawei Jiang, Beng Chin Ooi, Lei Shi and Sai Wu « The Performance of MapReduce: An In-depth Study », National University of Singapor.

[24] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, Bongki Moon, « Parallel Data Processing with MapReduce: A Survey ».

[25] Soila Kavulya, Jiaqi Tan, Rajeev Gandhi, Priya Narasimhan, « An Analysis of Traces from a Production MapReduce Cluster », December 2009 Parallel Data Laboratory University Pittsburgh.

[26] Xiufeng Liu, Christian Thomsen, Torben Bach Pedersen, « ETLMR: A Highly Scalable Dimensional ETL Framework based on MapReduce », August, 2011.

[27] Jonathan R.Owens Et Al, «Hadoop Real-World Solutions Cookbook ».

[28] ParAccel, « WhitePaper : Hadoop's Limitations for Big Data Analytics », 2012

[29] Tom White, « Hadoop the Definitive Guide », O'REILLY Yahoo Press

[30] Impetus Technologies, «WhitePaper : HADOOP PERFORMANCE TUNING », 2009

Bibliographie

2. Références webographiques

[W01] http://xpose.avenir.asso.fr/viewxpose.php?site=39&subpage=/intro.html

[W02] http://blog.cloudera.com/blog/2013/02/big-datas-new-use-cases-transformation-active-archive-and-exploration/

[W03] http://blog.octo.com/levolution-des-architectures-decisionnelles-avec-big-data/

[W04] https://www.redant.com/articles/big-data-crunching-the-numbers/

[W05] http://blog.trifork.com/2009/08/04/introduction-to-hadoop/

[W06]

[W07] http://fr.wikipedia.org

[W08] http://www.ubuntu-fr.org

[W09] http://www.libre-tic.com/systeme-decisionnels-sd.html

[W10] http://apache.org/

[W11] http://wiki.apache.org/hadoop

[W12] http://developer.yahoo.com/hadoop/tutorial

[W13] http://fredpalma.com/319/apache-hadoop/

[W14] http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

[W15] http://itsitspace.blogspot.com/2011/03/hadoop-compute-cluster-summary.html

Date de consultation : 06/07/2013.