N°d'ordre: .....

République Algerienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la Recherche scientifique

Université Djillali Liabès de Sidi Bel Abbès

Faculté des Sciences Exactes

Département d'informatique

Laboratoire EEDIS

Thèse de Doctorat

en Sciences

Filière : Informatique

Spécialité : Systèmes d'Information et de Connaissances

Par

Mr MERABET Mohamed

Un ordonnancement efficace des tâches pour les applications Big data

Soutenue le 23 Juin 2019. devant le jury :

Pr.	FARAOUN Mohamed Kamel	UDL SBA	Président du jury
Dr.	TOUMOUH Adil	UDL SBA	Examinateur
Dr.	AMAR BENSABER Djamel	ESI SBA	Examinateur
Dr.	KESKES Nabil	ESI SBA	Examinateur
Pr.	BENSLIMANE Sidi Mohammed	ESI SBA	Directeur de thèse
Dr.	GAFOUR Abdekader	UDL SBA	Co-Encadreur de thèse

Année Universitaire : 2018. - 2019.

A ma Mère et à mes frères Adel et Zine el Abidine

Remerciements

Je voudrais tout d abord exprimer mes plus profonds remerciements à ALLAH le tout-puissant, pour m'avoir accordé vie, santé et paix de l'esprit sans quoi je n'aurais pu achever ce travail ;

Je tiens à remercier le Président et les membres du jury de me faire l'honneur et l'immense privilège de juger mon travail.

Mes plus vifs remerciements sont adressés à mon encadreur Pr. Sidi Mohammed Benslimane pour m'avoir guidé, conseillé, soutenu tout au long des années de préparation de cette thèse. Je le remercie également pour m'avoir encouragé continuellement à progresser dans mon travail.

Ma sincère gratitude va à mes parents et mes frères pour leur présence et soutien indéfectible. Je leur exprime ici toute ma reconnaissance, en particulier ma mère.

Je conclurai en remerciant de tout cœur toutes les personnes qui auront contribué de près ou de loin à l'élaboration de cette thèse.

Sidi BEl Abbes , le 24 juin 2019.

Table des matières

Table des matières iv

Liste des figures vi

Liste des tableaux vii

Introduction Générale 1

0.1 Contexte 1

0.2 Problématique 2

0.3 Objectifs et contributions 3

0.4 Organisation 4

1 Big DATA : Caractéristiques et principaux défis 6

1.1 Introduction 6

1.2 Définitions 6

1.3 Histoire de big data 7

1.4 Caractéristiques 9

1.4.1 Vs de Big Data 9

1.4.2 Paradigmes de traitement 11

1.5 Architecture et cycle de vie de BIG-DATA 12

1.5.1 Phase I : Génération de données 13

1.5.2 Phase II : Acquisition de données 14

1.5.3 Phase III : Stockage des données 16

1.5.4 Phase IV : Analyse des données 25

1.6 Applications d'analyse de données Bigdata 26

1.6.1 Analyse de données structurées 26

1.6.2 Analyse du texte 27

1.6.3 Analyse du web 27

1.6.4 Analyse multimédia 29
1.6.5 Analyse de reseau 30
1.6.6 Analyse de mobile 30
1.7 Conclusion 32

2 Hadoop : Un Framework pour le traitement des données volumineuses et distribuées 34
2.1 Introduction 34
2.2 Architecture 34
2.2.1 Hadoop Distributed file system (HDFS) 35
2.2.2 MapReduce 37
2.3 Fonctionnement 39
2.3.1 Hadoop YARN 40
2.4 Caractéristiques 42
2.5 Problèmes et défis 42
2.6 Conclusion 45

3 Approches et techniques pour l'ordonnancement : Etat de l'art 46
3.1 Introduction 46
3.2 L'ordonnancement dans Hadoop : 46
3.2.1 Définition : 46
3.3 Les Exigences de Qualité d'ordonnancement 47
3.4 Classification des ordonnanceurs dans Hadoop 51
3.4.1 Ordonnancement des travaux 54
3.4.2 Ordonnancement des taches 63
3.5 Comparaison des approches d'ordonnancement 66
3.6 Conclusion 68

4 Un ordonnanceur prédictive de tâches map basé sur l'optimisation de la localité de données. 69
4.1 Introduction 69
4.2 Concepts clés du système 69
4.3 ARCHITECTURE DU SYSTÈME 72
4.3.1 L'ordonnanceur des tâches 73

4.3.2 Le gestionnaire de pré-chargement 74
4.3.3 L'analyseur de tâches 74
4.3.4 Temps d'exécution de la tâche map 74
4.3.5 Ordonnanceur des tâches 78
4.3.6 Pré-chargement de données : 80
4.4 Etudes expérimentales et validation 81
4.4.1 Environnement expérimental 81
4.4.2 Précision de prédiction du prédicteur de tâche 81
4.4.3 Localité de données 83
4.4.4 Temps d'exécution des tâches map 83
4.4.5 Discussion 84
4.5 Conclusion 85

Conclusion Générale et Perspectives **86**

Bibliographie **89**

Liste des figures

1.1 Un bref historique de l'evolution du Big Data [Hu et al., 2014]. 7
1.2 Quelques 'Vs'de Big Data [Assuncao et al., 2014]. 9
1.3 Variété de données [Assuncao et al., 2014]. 10
1.4 Vélocité des données [Assuncao et al., 2014]. 10
1.5 Plan technologique Big Data. [Hu et al., 2014] 12
1.6 Architecture réseau des systèmes de stockage [Hu et al., 2014]. 17
1.7 Modèle de données de Bigtable [Hu et al., 2014] 18
1.8 Modèle de données de Mongo DB [Hu et al., 2014]. 20

2.1 Architecture de système de fichiers distribués d'Hadoop 36
2.2 Architecture MapReduce d'Hadoop 38
2.3 Processus MapReduce d'Hadoop 39
2.4 Architecture YARN 40

3.1 Classification basée sur les exigences de qualité [Tiwari et al., 2015] . . . 47
3.2 Les entités MapReduce hiérarchiques pour être ordonnancé 51
3.3 Classification des ordonnanceurs des travaux hadoop [Usama et al., 2017] 55

4.1 L'architecture de système 73
4.2 La phase Map [Lin et al., 2012] 75
4.3 exemple d'ordonnancement. 78
4.4 La gestion du buffer 81
4.5 Prédiction de tâches map [Merabet et al., 2018] 82
4.6 Localité des données [Merabet et al., 2018] 83
4.7 Temps d'exécution total des tâches map [Merabet et al., 2018] 84

Liste des tableaux

1.1 Résumé des fonctionnalités des modèles de programmation[Hu et al., 2014]. 24
1.2 Taxonomie de l'analyse Big Data [Hu et al., 2014]. 32

3.1 Comparaison des approches d'ordonnancement. 68

4.1 Relation entre les étapes et les propriétés 76
4.2 La corrélation entre Tmap et les propriétés des différents travaux 76

Introduction Générale

0.1 Contexte

MapReduce a été largement déployé dans les grands clusters informatiques pour répondre au besoin croissant d'analyse des données de plus en plus volumineuses.

Hadoop est l'une des implémentations les plus populaires du framework MapReduce et a été adoptée par diverses organisations. Le framework MapReduce est implémenté sur des systèmes de fichiers distribués tels que le système de fichiers Google (GFS) et le système de fichiers distribué Hadoop (HDFS), qui divisent les grands ensembles de données en blocs de données et stockent plusieurs répliques.

Les travaux MapReduce sont soumis pour demander le traitement des données et chaque travail est divisé en plusieurs tâches map et tâches reduce. Une tâche map lit un bloc de données et le traite pour générer des résultats intermédiaires. Les tâches reduce permettant d'obtenir ces résultats intermédiaires et d'effectuer d'autres calculs pour obtenir les résultats finaux. Les tâches map et les tâches reduce sont affectées aux machines selon un algorithme d'ordonnancement. Lors de l'ordonnancement des tâches, une considération importante consiste à placer le calcul à proximité des données, c'est-à-dire à affecter une tâche sur ou à proximité de la machine qui stocke ses données d'entrée sur les disques locaux. Ceci est communément appelé le problème de la localité de données.

Le problème de la localisation des données est particulièrement crucial pour les tâches map car elles lisent les données du système de fichiers distribué et les fonctions map sont parallèles aux données. En outre, d'après une étude de trace empirique réalisée à partir d'un cluster MapReduce de production, la majorité des travaux sont map-intensive et beaucoup d'entre elles comporte uniquement des tâches map[Tiwari et al., 2015] ;[Hu et al., 2014]. Par conséquent, nous nous concentrons sur la localisation des données dans les algorithmes d'ordonnancement des tâches map et supposons que les

tâches reduce ne constituent pas un goulot d'étranglement du traitement des travaux ou du réseau de communication.

Dans la plupart des travaux existants, le lancement d'une tâche map sur une machine distante est considéré comme inefficace, car une machine distante doit d'abord extraire les données d'entrée d'autres machines via le réseau de communication avant de les traiter, ce qui introduit un délai supplémentaire à l'exécution de la tâche. Les tâches locales et distantes sont donc souvent modélisées avec des temps de traitement différents. Cette vue de la localité de données dans MapReduce est discutable. Si le réseau de communication qui connecte les machines avait une capacité infinie et pouvait transférer des données instantanément, il n'y aurait pas de différence entre l'affectation d'une tâche à ses machines locales ou à d'autres machines. Ainsi, le temps de traitement d'une tâche distante dépend de la capacité du réseau de communication et de l'algorithme d'ordonnancement qui alloue les tâches. Si les données peuvent être acheminées à l'avance pour que les machines n'attendent pas les données d'entrée avant d'exécuter les tâches, même si la capacité du réseau est limitée, nous pouvons toujours obtenir le même débit que si toutes les tâches étaient locales.

0.2 Problématique

L'ordonnancement de taches dans Hadoop présente une série de défis. Comme satisfaire des exigences de qualité (équité, localité de données, disponibilité, etc.) ou s'adapter à certaines contraintes liées à la configuration de l'infrastructure (hétérogénéité, saturation du réseau, etc.).

L'implémentation actuelle d'Hadoop n'a pas pris en compte ces exigences et contraintes. L'ordonnancement dans de tels cas peut engendrer une dégradation des performances des clusters Hadoop et causer des pannes.

Dans cette thèse, nous étudions le problème de la localisation des données et son impact sur l'ordonnancement de taches map, au-delà de la considération de la localité en trois niveaux de localité d'Hadoop. Nous adoptons une approche qui prend explicitement en compte la localisation des données lors d' ordonnancement et quantifie les limites fondamentales de la capacité d'un cluster Map Reduce.

Nous explorons aussi les algorithmes d'ordonnancement qui sont basés sur la localité des données pour réduire le temps totale d'exécution des travaux. Pour optimiser les performances d'ordonnancement des taches, nous sommes confrontés aux défis suivants :

Comment peut-on exploiter au mieux la localité des données ? L'exploitation de la localité des données dans Hadoop n'est pas optimale ce qui dégrade les performances et sature le réseau. Le pré-chargement des données est un mécanisme intéressant pour booster les performances des systèmes distribués. Il peut masquer la latence d'accès aux données en chevauchant le délai de transfert de données avec le traitement des données en même temps.

Comment peut-on déterminer avec précision où et quand déclencher le pré-chargement des données ? Nous allons construire un modèle de prédiction pour savoir l'enchainement des futures taches map et déclencher le pré-chargement des données requis par ces taches vers les nœuds d'exécution.

0.3 Objectifs et contributions

Nos contributions dans ce travail sont résumées comme suit :

1. Élaboration d'un état de l'art détaillé sur les différentes approches de l'ordonnancement des travaux et des taches, en particulier, ceux utilisant la localisation des données. Un tableau comparatif des différentes contributions selon des critères d'évaluation que nous avons définis, nous a permis d'une part de dégager les atouts et les limites de chaque approche, et d'autre part de mieux positionner notre approche pour voir comment il serait possible d'améliorer ce domaine de recherche.
2. Proposition d'une amélioration de l'architecture d'Hadoop pour capturer à la fois les emplacements des données d'entrée dans les nœuds du cluster et des informations sur les tâches en attente. Cette architecture fournit également des informations précises sur l'exécution des tâches en cours, permettant ainsi de prédire les nœuds où les prochaines taches map vont s'exécuter.
3. Utilisation de la technique de pré-chargement pour pré-charger les données d'entrées vers les nœuds là où les prochaines taches map seront affectées. Cela est

possible grâce à l'utilisation d'un modèle de prédiction basée sur les réseaux de neurones.

4. Proposition d'un algorithme d'ordonnancement des taches map basé sur le calcul de la localité de chaque nœud tout en prenant en compte la capacité de traitement de chaque nœud. L'algorithme utilise en entrée une liste des emplacements qui vont se libérer dans les nœuds pour effectuer un ordonnancement prédicatif des taches.
5. L'évaluation expérimentale des performances et de la précision de notre approche. Nous avons également étudié l'impact de notre approche sur la localisation des données. Enfin, nous avons comparé les performances du planificateur proposé avec le planificateur par défaut d' Hadoop.

0.4 Organisation

Cette thèse est organisée en quatre chapitres, à savoir :

Le *premier chapitre* expose le Contexte et la problématique de la thèse. ainsi que les contributions.

Dans le *deuxième chapitre* nous allons présenter le domaine de Big Data, le contexte historique de l'émergence de ce domaine et ces caractéristiques. Nous allons plus précisément décrire la chaîne de valeur qui comprend les étapes de génération, d'acquisition, de stockage et de traitement, ainsi que les types d'applications du Big Data selon leurs natures de données.

Dans le *troisième chapitre* nous décrivons l'architecture de base d' Hadoop, son fonctionnement, ses caractéristiques, ainsi que ses problèmes et défis.

Le *quatrième chapitre* dresse un état de l'art des principales approches et techniques pour l'ordonnancement. Dans ce chapitre, nous allons présenter une classification des ordonnanceurs des tâches dans l'environnement Hadoop. Enfin, nous terminons par une étude comparative entre les différentes approches existantes dans ce domaine.

Le *cinquième chapitre* introduit en détail notre ordonnanceur prédictif de tâches map avec pré-chargement des données. Nous présentons le modèle de prédiction basée sur les réseaux des neurones pour optimiser la localité de données et réduire ainsi le temps d'exécution des travaux.

Le *Chapitre conclusion et perspectives* met en avant les principales contributions et identifie les questions ouvertes et les perspectives de ce travail.

1 Big DATA : Caractéristiques et principaux défis

1.1 Introduction

Le Big Data est en train de devenir une nouvelle tendance technologique dans la science, l'industrie et les affaires [Gantz et Reinsel, 2012]. Le Big Data est en train de devenir lié à presque tous les aspects de l'activité humaine, depuis l'enregistrement des événements jusqu'à la recherche, la conception, la production et les services numériques. Inventé par les géants du web, au premier rang comme Yahoo, Google et Facebook, qui ont été les tous premiers à déployer ce type de technologie.

Le but de ce chapitre est de comprendre la nature du Big Data, ses principales caractéristiques, son évolution et d'étudier les composants qui constituent l'architecture du Big Data.

1.2 Définitions

Compte tenu de sa popularité actuelle, la définition du Big Data est plutôt diversifiée et il est difficile de parvenir à un consensus. Fondamentalement, le big data signifie non seulement un grand volume de données mais aussi d'autres caractéristiques qui le différencient des concepts de «données massives» et de «très grandes données»[Hu et al., 2014] .

International Data Corporation (IDC) est un pionnier dans l'étude du big data et de son impact [Gantz et Reinsel, 2011]. Il définit le Big data dans un rapport de 2011 comme : « Les technologies Big Data décrivent une nouvelle génération de technologies et d'architectures, conçues pour extraire économiquement la valeur a partir de

très gros volumes d'une large variété de données, en permettant une haute vélocité de capture, découverte et / ou d'analyse». Cette définition délimite les quatre principales caractéristiques des big data «4Vs», à savoir le volume, la variété, la vélocité et la valeur.

Le cabinet d'études Gartner en 2011 a noté que les défis et les opportunités de croissance des données sont tridimensionnels, c'est-à-dire qu'ils augmentent le volume, la vélocité et la variété. Une grande partie de l'industrie, y compris IBM et Microsoft utilisent ce modèle «3Vs» pour décrire big data.

1.3 Histoire de big data

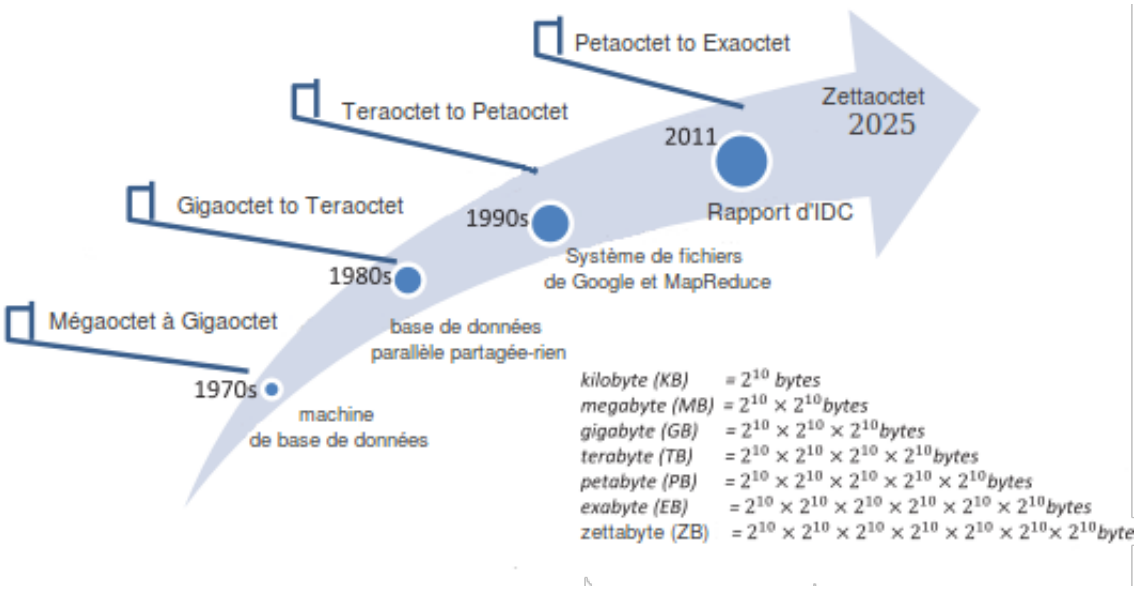

Figure 1.1 – *Un bref historique de l'evolution du Big Data [Hu et al., 2014].*

L'histoire du big data est étroitement liée à la capacité de stocker et gérer des ensembles de données de plus en plus volumineux, avec des limites de taille en expansion de plusieurs ordres de grandeur. La figure 1.1 montre l'histoire du Big Data en plusieurs étapes :

— **Mégaoctet à Gigaoctet :** Dans les années 1970 et 1980, les données commerciales historiques ont introduit le premier défi de «big data» en passant de mégaoctets à gigabits. Le besoin urgent à ce moment-là était d'héberger ces données et d'exécuter des requêtes relationnelles pour les analyses commerciales et les rapports. Ainsi, les systèmes de base de données sont des systèmes logiciels

qui imposent peu de contraintes sur le matériel et peuvent fonctionner sur des ordinateurs à usage général.

— **Gigaoctet to Teraoctet :** À la fin des années 1980, la popularisation de la technologie numérique a provoqué des volumes de données étendu à plusieurs gigaoctets ou même un téraoctet, ce qui est au-delà des capacités de stockage et / ou de traitement d'un seul grand système informatique. La parallélisation des données a été proposée pour étendre les capacités de stockage et améliorer les performances. Ainsi, plusieurs types de bases de données parallèles ont été construits, notamment des bases de données à mémoire partagée, des bases de données à disques partagés et des bases de données à rien partagé, l'architecture à rien partagé, construite sur un cluster en réseau de machines individuelles - chacune avec son propre processeur, mémoire et disque a connu un grand succès. Ce système exploite un modèle de données relationnel et des langages de requêtes relationnels déclaratifs, et utilise le parallélisme pour partitionner les données pour le stockage.

— **Teraoctet to Petaoctet :** À la fin des années 1990, c'est l'ère Internet, avec des données semi-structurés ou non structurés. Le besoin des sociétés de recherche était d'indexer et d'interroger le contenu croissant du Web. Malheureusement, les bases de données parallèles ne supportent pas bien les données non structurées. Pour gérer les données à l'échelle du Web, Google a créé le modèle de programmation Google File System (GFS) [Ghemawat et al., 2003] et MapReduce [Dean et Ghemawat, 2008]. GFS et MapReduce permettent la parallélisation automatique des données et la distribution d'applications de calcul à grande échelle à de grands groupes de serveurs. Au milieu des années 2000, Les bases de données NoSQL, qui sont sans schéma, rapides, hautement évolutives et fiables, ont commencé à émerger pour gérer ces données. En janvier 2007, Jim Gray, un pionnier de base de données, appele Big data le «quatrième paradigme». Il a également soutenu que la seule façon de faire face à ce paradigme était de développer une nouvelle génération d'outils informatiques pour gérer, visualiser et analyser le déluge de données.

— **Petaoctet to Exaoctet :** Selon les tendances de développement actuelles, les données stockées et analysées par les grandes entreprises atteindront sans aucun doute le petaoctet pour atteindre bientôt l'ampleur exaoctet. Cependant, la technologie actuelle gère encore les données téraoctets vers petaoctet. aucune technologie révolutionnaire n'a été développée pour faire face à de plus grands ensembles de données. En juin 2011, presque toutes les entreprises industrielles dominantes, y compris Oracle, Microsoft, Google, Amazon et Facebook, ont commencé à développer des projets big data. Cette initiative vise à favoriser le développement de méthodes avancées de gestion et d'analyse des données.

1.4 Caractéristiques

1.4.1 Vs de Big Data

Le Big Data est caractérisé par ce que l'on appelle souvent un modèle multi-V [Assuncao et al., 2014] , comme représenté sur la Fig. 1.2. La variété représente les types de données, la vélocité fait référence à la vitesse à laquelle les données sont produites et traitées et le volume définit la quantité de données. La véracité fait référence à la fiabilité des données compte tenu de la fiabilité de leur source, tandis que la valeur correspond à la valeur monétaire qu'une entreprise peut tirer de l'utilisation du Big Data.

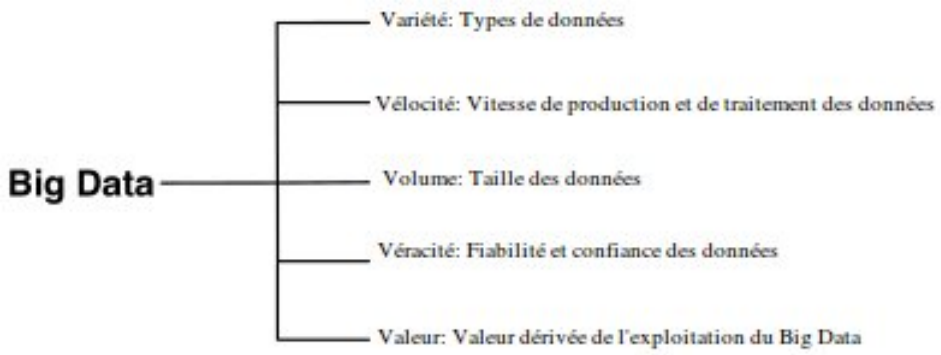

Figure 1.2 – *Quelques 'Vs'de Big Data [Assuncao et al., 2014].*

Les données sont également souvent disponibles à la vente sous la forme de rapports de recherche et techniques, de segment de marché et d'analyses financières, entre autres. Ces données peuvent être utilisées par diverses applications, par exemple, pour améliorer les conditions de vie dans les grandes villes, pour fournir des services de

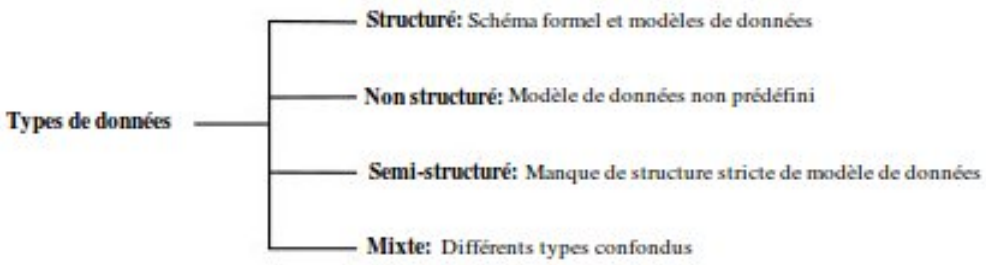

FIGURE 1.3 – *Variété de données [Assuncao et al., 2014].*

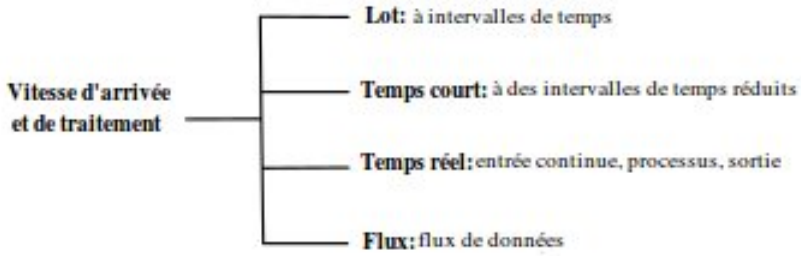

FIGURE 1.4 – *Vélocité des données [Assuncao et al., 2014].*

meilleure qualité, pour optimiser l'utilisation des ressources naturelles, et pour empêcher ou gérer la réponse aux événements imprévus.

La manipulation et l'analyse de ces données posent plusieurs défis car ils peuvent être de différents types (figure 1.3). Il est soutenu qu'une grande partie des données produites aujourd'hui est non structurée ou semi-structurée.

Considérant la vélocité des données, on remarque que, pour compliquer davantage les choses, les données peuvent arriver et nécessiter un traitement à différentes vitesses, comme illustré dans la figure 1.4. . Alors que pour certaines applications, l'arrivée et le traitement des données peuvent être effectués par lots, autres applications analytiques nécessitent des analyses continues et en temps réel, nécessitant parfois une action immédiate lors du traitement des flux de données entrants.

D'autres articles [DeVan, 2016] définit Big Data en sept V's en ajoutant la variabilité, la véracité, la visualisation et la valeur.
La variabilité est différente de la variété. Si le sens change constamment, cela peut avoir un impact considérable sur l'homogénéisation de vos données.
La véracité vise avant tout à ce que les données soient exactes, ce qui nécessite des processus permettant d'empêcher les mauvaises données de s'accumuler dans vos systèmes.
La visualisation est essentielle dans le monde d'aujourd'hui. L'utilisation de tableaux

et de graphiques pour visualiser de grandes quantités de données complexes est beaucoup plus efficace pour transmettre une signification que les feuilles de calcul et les rapports chargés de nombres et de formules.
La valeur est la fin du jeu. Après avoir abordé le volume, la vélocité, la variété, la variabilité, la véracité et la visualisation (ce qui prend beaucoup de temps, d'efforts et de ressources), vous voulez être sûr que votre organisation tire une valeur des données.

1.4.2 Paradigmes de traitement

L'analyse de Big Data peut être catégorisée en deux paradigmes alternatifs :

— **Traitement en continu :** Le paradigme de traitement en continu [Nesime, 2010] analyse les données le plus tôt possible pour en déduire les résultats. Dans ce paradigme, les données arrivent dans un flux. Dans son arrivée continue, parce que le flux est rapide et porte un volume énorme, seule une petite partie du flux est stockée dans une mémoire limitée. Le traitement en continu est utilisé pour les applications en ligne, généralement au deuxième niveau, voire à la milliseconde.

— **Traitement par lots :** Dans ce paradigme , les données sont d'abord stockées puis analysées. MapReduce [Dean et Ghemawat, 2008] est devenu le modèle de traitement par lots dominant. L'idée principale de MapReduce est que les données sont d'abord divisées en petits morceaux. Ensuite, ces morceaux sont traités en parallèle et de manière distribuée pour générer des résultats intermédiaires. Le résultat final est obtenu en agrégeant tous les résultats intermédiaires.

  Ce modèle planifie les ressources de calcul à proximité de l'emplacement des données, ce qui évite la surcharge de communication de la transmission de données. Récemment, la plupart des applications ont adopté le traitement par lots ; Même certaines applications de traitement en temps réel utilisent le traitement par lots pour obtenir une réponse plus rapide. De plus, des efforts de recherche ont été faits pour intégrer les avantages de ces deux paradigmes.

Les plates-formes Big Data peuvent utiliser des paradigmes de traitement alternatives ; cependant, les différences dans ces deux paradigmes entraîneront des distinctions architecturales dans les plateformes associées. Par exemple, les plates-formes

basées sur le traitement par lots englobent généralement des systèmes complexes de stockage et de gestion des données, tandis que les plates-formes basées sur le traitement en continu ne le font pas. En pratique, nous pouvons personnaliser la plateforme en fonction aux caractéristiques des données et aux exigences de l'application.

1.5 Architecture et cycle de vie de BIG-DATA

Dans cette section, nous nous concentrons sur la chaîne de valeur pour le big data. Plus précisément, nous décrivons une chaîne de valeur qui comprend quatre étapes (génération, acquisition, stockage et traitement). La Figure 1.5. représente les différentes phases dans le cycle de vie des données, Les détails de chaque phase sont expliqués comme suit.

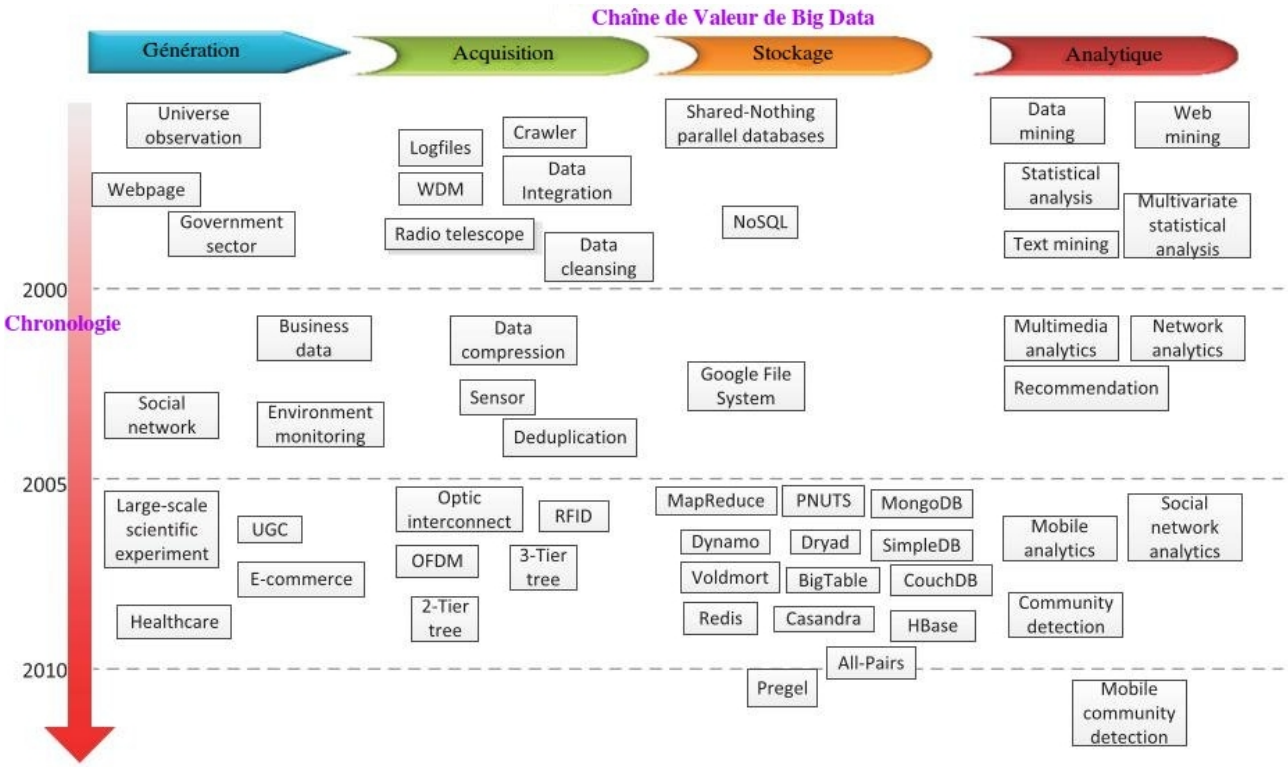

Figure 1.5 – *Plan technologique Big Data. [Hu et al., 2014]*

1.5.1 Phase I : Génération de données

Dans cette section, nous nous concentrons principalement sur trois sources types des big data :

1. **Données d'entreprises** L'utilisation des technologies de l'information et des données numériques a contribué à accroître la rentabilité du secteur des affaires pendant des décennies. Les transactions commerciales sur Internet, y compris les transactions entre entreprises et entre entreprises et consommateurs, atteindront 450 milliards par jour [John et David, 2010].

   Le volume toujours croissant de données métier nécessite une analyse en temps réel plus efficace pour obtenir des avantages supplémentaires. Par exemple, Amazon traite chaque jour des millions d'opérations et de requêtes de back-end auprès de plus d'un demi-million de vendeurs tiers [Julia, 2013].

2. **Données de réseaux** Les réseaux, comprenant Internet, le réseau mobile et l'Internet des objets, a pénétré dans les vies humaines sous tous les aspects possibles. Les applications réseau typiques, comme la recherche, les réseaux sociaux, les sites Web et les flux de clics. Ces sources génèrent des données à des vitesses record, exigeant des technologies avancées. Par exemple, Google, un moteur de recherche représentatif, traitait 20 petaoctet par jour en 2008 [Dean et Ghemawat, 2008]. Pour les applications de réseaux sociaux, Facebook a stocké, accédé et analysé plus de 30 petaoctet de données générées par les utilisateurs.

   Dans le domaine des réseaux mobiles, plus de 67% de la population mondiale, utiliseront des téléphones portables en 2019, et environ 57% d'entre eux vont avoir des smartphones [Statista, 2018]. Dans le domaine de l'Internet des objets, plus de 30 millions de nœuds de capteurs en réseau fonctionnent désormais dans les secteurs du transport, de l'automobile, de l'industrie, des services publics.

3. **Données scientifiques** nous mettons en évidence trois domaines scientifiques qui s'appuient de plus en plus sur l'analyse des big data :

   (a) **Biologie computationnelle :** Le Centre national pour l'innovation biotechnologique maintient la base de données GenBank de séquences nucléotidiques, qui contient plus de 250 milliards de bases nucléotidiques provenant de plus de 150 000 organismes distincts [Randal E., 2011].

   (b) **Astronomie :** le Sloane Digital Sky Survey (SDSS), le plus grand catalogue

astronomique, a généré 25 téraoctets de données provenant de télescopes. Comme les résolutions des télescopes ont augmenté, le volume de données généré chaque nuit devrait dépasser 20 téraoctets en 2014 [sdss].

(c) **Physique des hautes énergies :** L'expérience Atlas pour le grand collisionneur de hadrons au Centre de la recherche nucléaire européenne produira des données brutes à un rythme de 2 petaoctet par seconde au début de 2008 et stockera environ 10 petaoctet par an de données traitées [atlas].

1.5.2 Phase II : Acquisition de données

Le processus d'acquisition de données comprend deux sous-étapes, la collecte des données et le prétraitement des données. Dans cette sous section, nous allons voir ces trois sous-tâches.

COLLECTE DE DONNÉES

1. *Capteur :* Des capteurs sont couramment utilisés pour mesurer une quantité physique et la convertir en un signal numérique lisible pour traitement (et éventuellement stockage). Les types de capteurs comprennent le son, les vibrations, l'automobile, les produits chimiques, le courant électrique, Météo, la pression, le thermique et la proximité. Grâce aux réseaux filaires ou sans fil, ces informations peuvent être transférées vers un point de collecte de données.
2. *Fichiers journaux :* Les fichiers journaux, l'une des méthodes de collecte de données les plus largement déployées, sont générés par les systèmes de source de données pour enregistrer les activités dans un format de fichier spécifié en vue d'une analyse ultérieure. Les fichiers journaux sont utiles dans presque toutes les applications exécutées sur des appareils numériques. Par exemple, un serveur Web enregistre normalement tous les clics, accès et autres attributs effectués par un utilisateur de site Web dans un fichier journal d'accès.
3. *Explorateur web :* Un robot d'exploration est un programme qui télécharge et stocke des pages Web pour un moteur de recherche. En gros, un robot démarre avec un ensemble initial d'URL à visiter dans une file d'attente. Toutes les URL à extraire sont conservées et classées par ordre de priorité. À partir de cette file d'attente, le robot d'exploration obtient une URL ayant une certaine priorité, télécharge la page, identifie toutes les URL de la page téléchargée et ajoute les

nouvelles URL à la file d'attente. Ce processus est répété jusqu'à ce que le robot d'exploration décide d'arrêter. Les robots d'exploration Web sont des applications de collecte de données générales pour les applications Web, telles que les moteurs de recherche Web et les caches Web.

Prétraitement des données

Les techniques de prétraitement des données conçues pour améliorer la qualité des données devraient être en place dans les systèmes Big Data. Dans cette sous-section, nous examinons brièvement les efforts de recherche actuels pour trois techniques de prétraitement de données typiques.

1. *Intégration :* Les techniques d'intégration de données visent à combiner des données résidant dans différentes sources et à fournir aux utilisateurs une vue unifiée des données. L'intégration de données est un domaine mature dans la recherche de base de données traditionnelle [Korth et Silberschatz, 1986].

   Auparavant, deux approches prédominer, la méthode d'entrepôt de données et la méthode de fédération de données. La méthode d'entrepôt de données, également appelée ETL, comprend les étapes d'extraction, transformation et chargement.

   Deuxièmement, la méthode de fédération de données crée une base de données virtuelle pour interroger et agréger des données provenant de sources différentes. La base de données virtuelle ne contient pas de données mais contient à la place des informations ou des métadonnées sur les données réelles et leur emplacement.

2. *Nettoyage :* La technique de nettoyage des données fait référence au processus visant à déterminer des données inexactes, incomplètes ou irrationnels, puis à modifier ou supprimer ces données pour améliorer la qualité des données.

   Le nettoyage des données est généralement considéré comme essentiel pour garder les données cohérentes et mises à jour. Par exemple, Dans le domaine du commerce électronique, bien que la plupart des données soient collectées électroniquement, il peut y avoir de sérieux problèmes de qualité des données comme les bogues logiciels, les erreurs de personnalisation et le processus de configuration du système.

3. *La redondance des données :* La redondance des données est la répétition ou la superflue des données, ce qui est un problème commun pour divers ensembles de données. La redondance des données augmente inutilement le débit de transmission des données et entraîne des inconvénients pour les systèmes de stockage, comprenant le gaspillage d'espace de stockage , l'incohérence des données, une fiabilité réduite et la corruption des données.

1.5.3 Phase III : Stockage des données

Le sous-système de stockage de données dans une plateforme big data organise les informations collectées dans un format pratique pour l'analyse et l'extraction de valeur. À cette fin, le sous-système de stockage de données devrait fournir deux ensembles de caractéristiques :

— L'infrastructure de stockage doit prendre en charge les informations de manière persistante et fiable.

— Le sous-système de stockage de données doit fournir une interface d'accès évolutive permettant d'interroger et d'analyser une grande quantité de données.

Cette décomposition fonctionnelle montre que le sous-système de stockage de données peut être divisé en infrastructure matérielle et gestion des données. Ces deux composants sont expliqués ci-dessous.

Infrastructure de stockage

L'infrastructure matérielle est responsable du stockage physique des informations collectées. les dispositifs de stockage sont classés généralement en fonction d'architecture de réseau [Ulf et al., 2011], *Stockage Directement Connectée (DAS)*, *Stockage Connectés en Réseau (NAS)*, *Réseau de zone de stockage (SAN)*.

L'architecture de réseau de ces trois technologies est illustrée à la Figure. 1.6. Le système SAN possède l'architecture la plus compliquée, en fonction des périphériques réseau spécifiques.

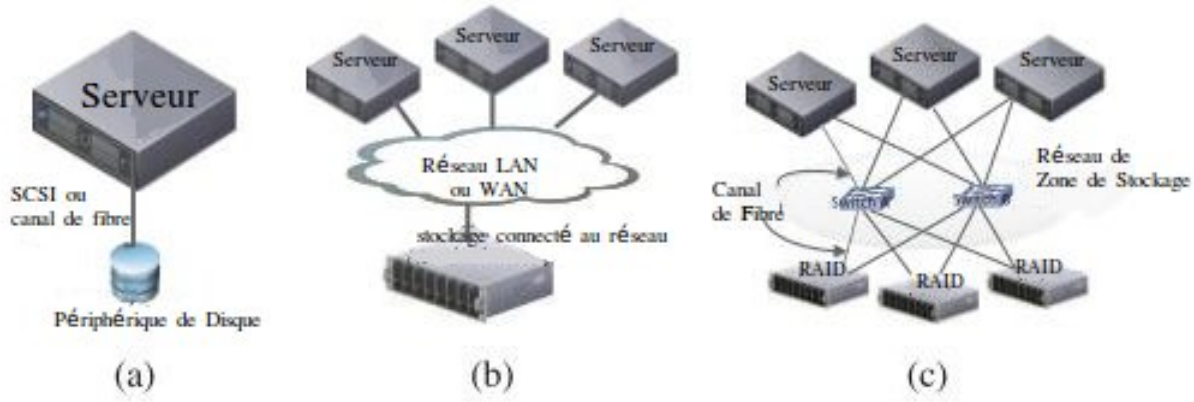

FIGURE 1.6 – *Architecture réseau des systèmes de stockage [Hu et al., 2014].*

Framework de gestion des données

Dans cette section, nous allons classer les frameworks de gestion des données en trois couches constituées de systèmes de fichiers, de technologies de base de données et de modèles de programmation.

Systèmes de fichiers : Le système de fichiers est à la base du stockage des big data et attire donc une grande attention de la part de milieu académique et industriel. Google a conçu et mis en œuvre GFS en tant que système de fichiers distribués évolutif pour les grandes applications distribuées à haute densité de données. GFS fonctionne sur des serveurs peu onéreux pour offrir une tolérance aux pannes et des performances élevées à un grand nombre de clients.

D'autres entreprises et chercheurs ont aussi développé leurs propres solutions pour répondre aux besoins spécifiques de stockage de Big Data. HDFS [hdfs] et Kosmosfs [Kosmosfs] sont des dérivés open source de GFS. Microsoft a créé Cosmos [Chaiken et al., 2008] pour soutenir ses activités de recherche et de publicité. Facebook a mis en place Haystack [Beaver et al., 2010] pour stocker une quantité massive de photos en petits fichiers.

Technologies de bases de données : Plusieurs systèmes de bases de données ont été proposés pour différentes échelles d'ensembles de données et diverses applications. Les systèmes de bases de données relationnelles traditionnels ne peuvent évidemment pas répondre aux défis de variété et d'échelle requis par les big data. En raison de certaines caractéristiques indispensables, notamment être sans schéma, prise en charge

de la réplication facile, la simplicité de l'API, cohérence et la prise en charge d'une grande quantité de données, la base de données NoSQL devient la norme pour gérer les problèmes des big data.

Dans cette sous-section, nous nous concentrons principalement sur trois principaux types de bases de données NoSQL organisées par le modèle de données, à savoir les stockages clés-valeurs, les bases de données en colonnes et les bases de documents.

1. *Stockages clé-valeur :* les stockages clés-valeurs ont un modèle de données simple dans lequel les données sont stockées en tant que paire clé-valeur. Chacune des clés est unique et les clients mettent ou demandent des valeurs pour chaque clé.

   Les bases de données de valeur-clé qui ont émergé ces dernières années ont été fortement influencées par Dynamo d'Amazon [DeCandia et al., 2007]. Dans Dynamo, les données doivent être partitionnées sur un cluster de serveurs et répliquées sur plusieurs copies. L'évolutivité et la durabilité reposent sur deux mécanismes clés : le partitionnement et la réplication et le contrôle de version des objets.

2. *Bases de données orientées colonne :* Les bases de données orientées colonne stockent et traitent les données par colonne plutôt que par ligne. Les lignes et les colonnes seront réparties sur plusieurs nœuds afin de garantir l'évolutivité. La principale source d'inspiration pour les bases de données orientées colonne est le Bigtable de Google, qui sera discuté en premier, suivi de plusieurs dérivés.

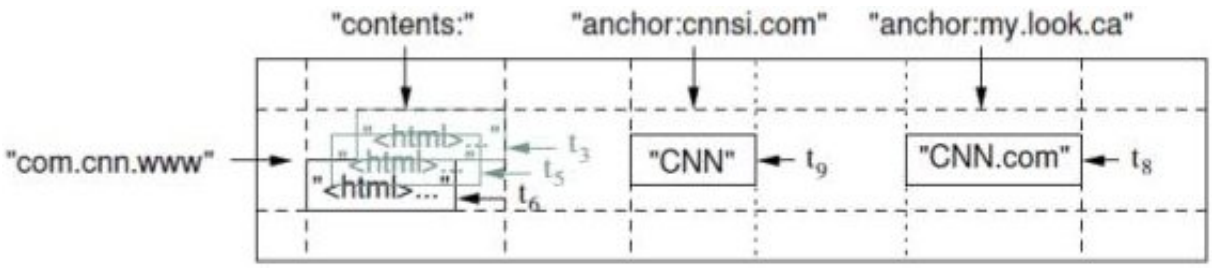

FIGURE 1.7 – *Modèle de données de Bigtable [Hu et al., 2014]*

— **BigTable :** [Chang et al., 2008] : La structure de données de base de Bigtable est un plan trié multidimensionnel, distribuée et persistante. Le plan est indexé par une clé ligne, une clé colonne et un horodatage. Les lignes sont conservées

dans l'ordre lexicographique et sont partitionnées dynamiquement en tablettes, qui représentent l'unité de distribution et l'équilibrage de charge. Les colonnes sont regroupées par leur préfixe de clé dans des ensembles appelés familles de colonnes qui représentent l'unité de base du contrôle d'accès. Un horodatage est utilisé pour différencier les réversions d'une valeur de cellule.

La figure 1.7 illustre un exemple de stockage d'une grande collection de pages Web dans une table unique, dans laquelle des URL sont utilisées en tant que clés de ligne et divers aspects de pages Web sont utilisés en tant que noms de colonnes. Le contenu des pages Web associées à plusieurs versions est stocké dans une seule colonne.

— **Cassandra :** [Lakshman et Malik, 2009] : développée par Facebook et devenu open source en 2008, rassemble les technologies de systèmes distribués de Dynamo et le modèle de données de Bigtable.

  En particulier, une table dans Cassandra est un plan distribué multidimensionnel structuré en quatre dimensions : les lignes, les familles de colonnes, les colonnes et les super colonnes. Les mécanismes de partition et de réplication de Cassandra sont similaires à ceux de Dynamo, ce qui garantit une éventuelle cohérence.

— **Dérivés de Bigtable :** comme le code Bigtable n'est pas disponible sous licence open source, des projets open source, tels que HBase [hbase, 2013] et Hypertable [hypertable, 2013], ont émergé pour implémenter des systèmes similaires en adoptant les concepts décrits dans la sous-section Bigtable.

Les bases de données orientées colonne sont similaires car elles sont principalement modelées après Bigtable mais diffèrent par des mécanismes de concurrence et d'autres fonctionnalités. Par exemple, Cassandra se concentre sur une faible concurrence via le contrôle de simultanéité multiversion, tandis que HBase et HyperTable se concentrent sur une forte cohérence via les verrous et la journalisation.

Bases de données documentaires : Les stockages documents prennent en charge des structures de données plus complexes que les stockages de valeurs-clés. Il n'y a pas de schéma strict auquel les documents doivent se conformer, ce qui élimine le besoin

d'efforts de migration de schéma. MongoDB, SimpleDB et CouchDB sont les trois principaux représentants. Les modèles de données de tous les stockages documents ressemblent à l'objet JSON [Crochford, 2006].

La figure 1.8 montre un article wiki représenté dans le format de document MongoDB [mongoDb, 2013]. Les principales différences dans les stockages documents résident dans les mécanismes de réplication et de cohérence des données.

```
{
  title: "MongoDB",
  last_editor: "172.5.123.91",
  last_modified: new Date("9/23/2013"),
  body: "MongoDb is a ..."
  categories: ["Database", "NoSQL", "Document Databases"],
  reviewed: false
}
```

FIGURE 1.8 – *Modèle de données de Mongo DB [Hu et al., 2014].*

Modèles de programmation Bien que les bases de données NoSQL soient intéressantes pour de nombreuses raisons, contrairement aux systèmes de bases de données relationnelles, elles ne prennent pas en charge l'expression déclarative de l'opération de jointure et offrent une prise en charge limitée des opérations d'interrogation et d'analyse. Le modèle de programmation est essentiel à la mise en œuvre des logiques applicatives et à la facilitation des applications d'analyse de données. Cependant, il est difficile pour les modèles parallèles traditionnels comme MPI d'implémenter des programmes parallèles sur une échelle de big data, c'est-à-dire des centaines ou même des milliers de serveurs sur une large zone. De nombreux modèles de programmation parallèle ont été proposés pour résoudre des applications spécifiques à un domaine.

Ces modèles efficaces améliorent les performances des bases de données NoSQL et réduisent l'écart de performances avec les bases de données relationnelles. Les bases de données NoSQL sont déjà en train de devenir la pierre angulaire de l'analyse massive des données.

Nous discutons en particulier de trois types de modèles : le modèle de traitement générique, le modèle de traitement de graphe et le modèle de traitement de flux.

— **Modèle de traitement générique :** Ce type de modèle traite des problèmes d'application généraux utilisé dans MapReduce [Dean et Ghemawat, 2008] et

ses variantes, et dans Dryad [Isard et al., 2007]. MapReduce est un modèle de programmation simple et puissant qui permet la mise en parallèle automatique et la distribution d'applications de calcul à grande échelle sur de grands groupes de PC. Le modèle de calcul est constitué de deux fonctions définies par l'utilisateur, appelées Map et Reduce.

L'infrastructure MapReduce regroupe toutes les valeurs intermédiaires associées à la même clé intermédiaire I et les transmet à la fonction Reduce. La fonction Reduce reçoit une clé intermédiaire et son ensemble de valeurs et les fusionne pour générer un ensemble de valeurs (généralement) plus petit.

Le framework MapReduce fournit uniquement deux fonctions opaques, sans certaines des opérations les plus courantes (par exemple, la projection et le filtrage). L'ajout de la syntaxe SQL au-dessus de l'infrastructure MapReduce est un moyen efficace de rendre MapReduce facile à utiliser pour les programmeurs de base de données traditionnels qualifiés en SQL.

Dryad est un moteur d'exécution distribuée à usage général pour les applications parallèles de données. Un travail Dryad est un graphe acyclique orienté dans lequel chaque sommet est un programme et les bords représentent des canaux de données. Dryad exécute le travail en exécutant les sommets de ce graphe sur un ensemble d'ordinateurs, en communiquant via des canaux de données, comprenant des fichiers, des canaux TCP et des FIFO à mémoire partagée.

Le graphe de calcul logique est automatiquement mappé sur les ressources physiques a l'éxécution. Le modèle de programmation MapReduce peut être considéré comme un cas particulier de Dryad dans lequel le graphe se compose de deux étapes : les sommets de l'étape Map mélangent leurs données aux sommets de l'étape reduce.

— **Modèle de traitement de graphe :** Une classe croissante d'applications (par exemple, l'analyse de réseau social, RDF) peut être exprimée en termes d'entités liées les unes aux autres et capturées en utilisant des modèles en graphe. Contrairement aux modèles de type flux, le traitement de graphe itératif par nature et le même jeu de données peuvent être revisité plusieurs fois.

Nous considérons principalement Pregel [Malewicz et al., 2010] et GraphLab

[Low et al., 2012]. Pregel de Google se spécialise dans le traitement de graphe à grande échelle, comme l'analyse de graphe Web et de réseaux sociaux. La tâche de calcul est exprimée sous la forme d'un graphe orienté, constitué de sommets et d'arêtes orientés. Chaque sommet est associé à une valeur modifiable et définie par l'utilisateur. Les arêtes orientées sont associées à leurs sommets source, et chaque arête est constituée d'une valeur modifiable et d'un identificateur de sommet cible. Après l'initialisation du graphe, les programmes sont exécutés comme une séquence d'itérations, appelées supersteps, qui sont séparées par des points de synchronisation globaux jusqu'à ce que l'algorithme se termine avec la sortie. Au sein de chaque superstep, les vertices exécutent en parallèle la même fonction définie par l'utilisateur qui exprime la logique d'un algorithme donné. Un sommet peut modifier son état ou celui de ses arêtes sortants, recevoir des messages qui lui sont transmis dans le précédent superstep, envoyer des messages à d'autres sommets, ou même muter la topologie du graphe. Une arête n'a pas de calcul associé. Un sommet peut se désactiver en votant pour s'arrêter. Lorsque tous les sommets sont simultanément inactifs et qu'aucun message n'est en transit, le programme entier se termine. Le résultat d'un programme Pregel est l'ensemble des valeurs de sortie des sommets, ce qui est souvent un graphe orienté isomorphe à l'entrée.

GraphLab est un autre modèle de traitement de graphe qui cible les algorithmes parallèle d'apprentissage automatique. L'abstraction de GraphLab comprend trois composants : le graphe de données, la fonction de mise à jour et l'opération de synchronisation. Le graphe de données est un conteneur qui gère les données définies par l'utilisateur, y compris les paramètres du modèle, l'état de l'algorithme et même les données statistiques. La fonction de mise à jour est une procédure sans état qui modifie les données dans la portée d'un sommet et planifie l'exécution future des fonctions de mise à jour sur d'autres sommets. Enfin, l'opération de synchronisation maintient simultanément les agrégats globaux.

La différence clé entre Pregel et GraphLab se trouve dans leurs modèles de synchronisation. Pregel a une barrière à la fin de chaque itération et tous les sommets doivent atteindre un état de synchronisation global après itération, alors que GraphLab est complètement asynchrone, conduisant à des sommets

plus complexes. GraphLab propose trois modèles de cohérence, la cohérence totale, arête et sommet, pour permettre différents niveaux de parallélisme.

— **Modèle de traitement de flux :** S4 [Neumeyer et al., 2010] et Storm [storm, 2013] sont deux plates-formes de traitement de flux réparties qui s'exécutent sur la machine virtuelle Java.

  S4 implémente le modèle de programmation des acteurs. Chaque tuple à clé du flux de données est traité comme un événement et routé avec une affinité avec les éléments de traitement (PE). Les PE forment un graphe acyclique dirigé et prennent en charge le traitement des événements avec certaines clés et publient les résultats. Les noeuds de traitement (PN) sont les hôtes logiques des PE et sont chargés d'écouter les événements et de transmettre les événements entrants au conteneur d'éléments de traitement (PEN), qui invoque les PE appropriés dans l'ordre approprié.

  Storm partage de nombreuses fonctionnalités avec S4. Un travail Storm est également représenté par un graphe orienté et sa tolérance aux fautes est partielle à la suite du canal de streaming entre les sommets. La principale différence entre S4 et Storm est l'architecture : S4 adopte une architecture décentralisée et symétrique, tandis que Storm est un système maître-esclave comme MapReduce.

	MapReduce	**Dryad**	**Pregel**	**GraphLab**	**Storm**	**S4**
Application	moteur d'exécution parallèle polyvalent	moteur d'exécution parallèle polyvalent	traitement de graphe à grande échelle	Apprentissage automatique et exploration de données à grande échelle	traitement en streaming distribué	traitement en streaming distribué
Modèle de Programmation	Map et Reduce	graphe orienté acyclique	graphe orienté	graphe orienté	graphe orienté acyclique	graphe orienté acyclique
Parallélisme	exécution simultanée au sein des phases Map et Reduce	exécution simultanée de sommets au cours d'une étape	exécution simultanée sur les sommets d'un superstep	exécution simultanée des champs non chevauchantes définies par un modèle de cohérence	processus de travail et exécuteurs	processus de travail et exécuteurs
Traitement des Données	système de fichiers distribué	divers supports de stockage	système de fichiers distribué	mémoire ou disque	mémoire	mémoire
Architecture	maîtres-esclaves	maîtres-esclaves	maîtres-esclaves	maîtres-esclaves	maîtres-esclaves	décentralisé et symétrique
Tolérance aux Pannes	tolérance aux pannes au niveau du noeud	tolérance aux pannes au niveau du noeud	points de reprise	points de reprise	tolérance partielle aux pannes	tolérance partielle aux pannes

TABLE 1.1 – *Résumé des fonctionnalités des modèles de programmation[Hu et al., 2014].*

Le tableau 1.1 montre une comparaison de caractéristiques des modèles de programmation discutés ci -dessus.

Premièrement, bien que le traitement en temps réel devient plus important, le traitement par lots reste le paradigme le plus courant du traitement des données.

Deuxièmement, la plupart des systèmes adoptent le graphe comme modèle de programmation parce que le graphe peut exprimer des tâches plus complexes.

Troisièmement, tous les systèmes prennent en charge l'exécution simultanée pour accélérer la vitesse de traitement.

Quatrièmement, les modèles de traitement en continu utilisent la mémoire comme support de stockage de données pour atteindre des taux d'accès et de traitement plus élevés, tandis que les modèles de traitement par lots utilisent un système de fichiers ou un disque pour stocker des données volumineuses.

Cinquièmement, l'architecture de ces systèmes est typiquement maître-esclave ; Cependant, S4 adopte une architecture décentralisée.

Enfin, la stratégie de tolérance aux pannes est différente pour différents systèmes. Pour Storm et S4, lorsqu'une défaillance de nœud se produit, les processus sur le nœud défaillant sont déplacés vers des nœuds de secours. Pregel et GraphLab utilisent le point de reprise pour la tolérance aux pannes, qui est invoqué au début de certaines itérations. MapReduce et Dryad ne prennent en charge que la tolérance aux pannes au niveau du noeud.

1.5.4 Phase IV : Analyse des données

L'analyse de données concerne les informations obtenues par l'observation, la mesure ou les expériences sur un phénomène d'intérêt. L'objectif de l'analyse de données est d'extraire le maximum d'informations pertinentes au sujet considéré. La nature du sujet et le but peuvent varier considérablement. Les listes suivantes ne mentionnent que quelques objectifs potentiels :

- Pour extrapoler et interpréter les données et déterminer comment les utiliser,
- Pour vérifier si les données sont rational,
- Pour donner des conseils et aider à la prise de décision,
- Pour diagnostiquer et déduire les raisons d'un défaut.
- Pour prédire ce qui se passera dans le futur.

Nous pouvons classer les données en trois niveaux en fonction de la profondeur de l'analyse : l'analyse descriptive, l'analyse prédictive et l'analyse prescriptive[Gavin, 2013].

- **Analyse Descriptive :** exploite les données historiques pour décrire ce qui s'est passé. Par exemple, une régression peut être utilisée pour trouver des tendances simples dans les ensembles de données, la visualisation présente les données de manière significative, et la modélisation des données est utilisée

pour collecter, stocker et découper les données de manière efficace. Les analyses descriptives sont généralement associées à business intelligence des systèmes de veille économique ou systèmes de visibilité.

— **Analyses Prédictives :** met l'accent sur la prévision des probabilités et des tendances futures. Par exemple, la modélisation prédictive utilise des techniques statistiques telles que la régression linéaire et logistique pour comprendre les tendances et prédire les résultats futurs, et data mining extrait des modèles pour fournir des informations et des prévisions.

— **Analyse Prescriptive :** abord les problèmes de la prise de décision et de l'efficacité. Par exemple, la simulation est utilisée pour analyser des systèmes complexes afin de mieux comprendre le comportement du système et d'identifier les problèmes, et des techniques d'optimisation sont utilisées pour trouver des solutions optimales sous des contraintes données.

1.6 Applications d'analyse de données Bigdata

Dans cette section nous discutons six types d'applications Big Data, organisées par type de données : l'analyse de données structurée, l'analyse de texte, l'analyse Web, l'analyse multimédia, l'analyse de réseau et l'analyse mobile.

1.6.1 Analyse de données structurées

Une grande quantité de données structurées est générée dans les domaines des affaires et de la recherche scientifique. La gestion de ces données structurées repose sur le SGBDR mature, l'entreposage de données. L'analyse de données est largement fondée sur l'exploration de données et l'analyse statistique, comme décrit ci-dessus. Ces deux domaines ont été minutieusement étudiés au cours des trois dernières décennies. Récemment, l'apprentissage en profondeur, un ensemble de méthodes d'apprentissage automatique basées sur des représentations d'apprentissage, devient un domaine de recherche actif.

La plupart des algorithmes d'apprentissage automatique actuels dépendent de représentations et de caractéristiques d'entrée conçues par l'homme, ce qui est une tâche

complexe pour diverses applications. Les algorithmes d'apprentissage en profondeur intègrent l'apprentissage par représentation et apprennent plusieurs niveaux de représentation de complexité / abstraction.

En outre, de nombreux algorithmes ont été appliqués avec succès aux applications émergentes. L'apprentissage statistique, basé sur des modèles mathématiques précis et des algorithmes puissants, a déjà été appliqué à la détection d'anomalies et au contrôle de l'énergie. En utilisant des caractéristiques de données, l'exploration temporelle et spatiale pour extraire des structures de connaissances représentées dans des modèles pour des flux de données à grande vitesse et des données de capteurs. Motivée par les préoccupations en matière de protection de la vie privée dans les applications de commerce électronique, d'administration en ligne et de soins de santé, la préservation des données confidentielles devient un domaine de recherche actif. Au cours de la dernière décennie, en raison de la disponibilité croissante de données d'événement et de techniques de découverte de processus et de vérification de conformité, l'exploration de processus [van der Aalst, 2012] est devenue un nouveau domaine de recherche.

1.6.2 Analyse du texte

Le texte est l'une des formes les plus courantes d'informations stockées et comprend la communication par e-mail, les documents d'entreprise, les pages Web et le contenu des médias sociaux. Par conséquent, l'analyse de texte est considérée comme ayant un potentiel commercial plus élevé que l'exploration de données structurée.

En général, l'analyse de texte, également appelée exploration de texte, fait référence au processus d'extraction d'informations et de connaissances utiles à partir d'un texte non structuré. L'exploration de texte est un domaine interdisciplinaire à la croisée de la recherche documentaire, de l'apprentissage automatique, des statistiques, de la linguistique computationnelle et, en particulier, de l'exploration de données. La plupart des systèmes d'exploration de texte reposent sur la représentation textuelle et le traitement du langage naturel (NLP).

1.6.3 Analyse du web

L'analyse Web vise à récupérer, extraire et évaluer automatiquement les informations pour la découverte des connaissances à partir de documents Web et de services.

Ce champ est construit sur plusieurs domaines de recherche, y compris les bases de données, la recherche d'information, le TLN et l'exploration de texte.

Nous pouvons catégoriser l'analyse Web en trois domaines d'intérêt en fonction de la partie du site Web exploitée : l'exploration de contenu Web, l'exploration de structure Web et l'exploration de l'utilisation Web [Sankar K. et al., 2002].

L'exploration de contenu Web est la découverte d'informations ou de connaissances utiles à partir du contenu du site Web. Étant donné que la plupart des données de contenu Web sont des données textuelles non structurées, une grande partie de l'effort de recherche est centré sur le texte et le contenu hypertexte. L'exploration de texte est un sujet bien développé, tel que décrit ci-dessus. L'exploration hypertexte implique l'exploration de pages HTML semi-structurées comportant des liens hypertexte.

L'apprentissage supervisé ou la classification joue un rôle clé dans l'exploration de l'hypertexte, notamment dans la gestion des courriels, la gestion des groupes de discussion et la gestion des répertoires Web [Chakrabarti, 2000].

L'exploration de structure Web est la découverte du modèle sous-jacent des structures de liens sur le Web. Ici, la structure représente le graphe des liens dans un site ou entre des sites. Le modèle est basé sur la topologie du lien hypertexte avec ou sans description de lien. Ce modèle révèle les similitudes et les relations entre les différents sites Web et peut être utilisé pour classer les sites Web. Les méthodes Page Rank et CLEVER exploitent ce modèle pour trouver des pages Web. La recherche ciblée [Chakrabarti et al., 1999] est un autre exemple qui utilise avec succès ce modèle. L'objectif d'un robot est de rechercher sélectivement des sites Web liés à un ensemble prédéfini de sujets. Plutôt que de collecter et d'indexer tous les documents Web accessibles, un robot ciblé analyse sa limite d'exploration pour trouver les liens les plus pertinents pour l'exploration et évite les régions non pertinentes du Web, ce qui permet d'économiser du matériel et des ressources réseau importants.

L'exploration de l'utilisation Web fait référence à l'extraction de données secondaires générées par des sessions Web ou des comportements. L'exploration de l'utilisation du Web diffère de l'exploration de contenu Web et de l'exploration de la structure Web, qui utilisent les données réelles ou primaires sur le Web. Les données d'utilisation Web incluent les données des journaux d'accès du serveur Web, des journaux du navigateur, des profils utilisateur, des données d'enregistrement, des sessions utilisateur

ou des transactions, des cookies, des requêtes utilisateur, des signets, des clics et toute autre donnée générée par l'interaction des utilisateurs et du web.

L'exploration de l'utilisation du Web joue un rôle important dans le commerce électronique, la confidentialité / sécurité du Web et de plusieurs autres domaines émergents. Par exemple, les systèmes de recommandation collaborative permettent la personnalisation pour le commerce électronique en exploitant les similitudes et les dissimilarités dans les préférences des utilisateurs [Xu et al., 2012].

1.6.4 Analyse multimédia

Récemment, les données multimédia, y compris l'image, l'audio et la vidéo, ont augmenté à un rythme phénoménal et sont presque omniprésentes. L'analyse de contenu multimédia fait référence à l'extraction de connaissances et à la compréhension de la sémantique capturée dans les données multimédias. Parce que les données multimédias sont diverses et plus riches en informations que les simples données structurées et textuelles dans la plupart des domaines, l'extraction d'informations implique de combler le fossé sémantique des données multimédia. La recherche en analyse multimédia couvre un large éventail de sujets, y compris la synthèse multimédia , l'annotation multimédia, l'indexation et la récupération multimédia, la recommandation multimédia et la détection d'événements multimédias.

L'annotation multimédia fait référence à l'attribution d'images et de vidéos à un ensemble d'étiquettes décrivant leur contenu aux niveaux syntaxique et sémantique. Avec l'aide de ces étiquettes, la gestion, la synthèse et la récupération de contenu multimédia peuvent être réalisées facilement. Parce que l'annotation manuelle prend du temps et nécessite des coûts de main-d'œuvre élevés. L'indexation et la récupération multimédias concernent la description, le stockage et l'organisation de l'information multimédia pour aider les utilisateurs à trouver des ressources multimédias facilement et rapidement [Hu et al., 2011].

L'objectif de la recommandation multimédia est de proposer des contenus multimédias spécifiques à un utilisateur en fonction des préférences de l'utilisateur, ce qui a été prouvé comme une méthode efficace pour fournir une personnalisation de haute qualité. La plupart des systèmes de recommandation actuels sont basés soit sur le filtrage de contenu ou sur le filtrage collaboratif. Les approches basées sur le contenu iden-

tifient les caractéristiques communes de l'intérêt des utilisateurs et recommandent à l'utilisateur un autre contenu qui partage des fonctionnalités similaires. Ces approches reposent entièrement sur des mesures de similarité de contenu et souffrent des problèmes d'analyse de contenu limitée et de sur-spécification. Les approches fondées sur le filtrage collaboratif identifient le groupe de personnes qui partagent des intérêts communs et recommandent le contenu en fonction du comportement des autres membres du groupe.

1.6.5 Analyse de reseau

En raison de la croissance rapide des réseaux sociaux en ligne, l'analyse de réseau a évolué a partir de l'analyse bibliométrique et de l' analyse de réseau de sociologie à l'analyse de réseaux sociaux émergents du début des années 2000. Typiquement, les réseaux sociaux contiennent énormément de données de liaison et de contenu, où les données de liaison sont essentiellement la structure graphique, représentant les communications entre entités et les données de contenu contenant du texte, des images et d'autres données multimédia dans les réseaux. Évidemment, la richesse des données de réseaux sociaux offre des défis et des opportunités sans précédent pour l'analyse de données.

En raison du développement révolutionnaire de la technologie Web 2.0, le contenu généré par les utilisateurs explose sur les réseaux sociaux. Le terme médias sociaux est employé pour nommer un tel contenu généré par les utilisateurs, y compris les blogs, les microblogs, le partage de photos et de vidéos, le marketing social de livres, les sites de réseautage social, les actualités sociales et les wikis. Le contenu des médias sociaux contient du texte, du multimédia, des emplacements et des commentaires. Presque tous les sujets de recherche sur l'analyse de données structurées, l'analyse de texte et l'analyse multimédia peuvent être traduits en analyse de médias sociaux.

1.6.6 Analyse de mobile

L'énorme volume d'applications et de données conduit à l'émergence de l'analyse mobile ; Cependant, l'analyse mobile est confrontée à des défis causés par les caractéristiques inhérentes aux données mobiles, telles que la sensibilisation aux mobiles, la sensibilité aux activités, le bruit et la richesse de la redondance. Actuellement, l'analyse

mobile est loin d'être mature. Ainsi, nous étudions seulement certaines des applications d'analyse les plus récentes et représentatives.

Le RFID permet à un capteur de lire un code d'identification de produit (EPC) unique associé à une étiquette à distance [Han et al., 2010]. Les étiquettes peuvent être utilisées pour identifier, localiser, suivre et surveiller des objets physiques de manière rentable. Actuellement, la RFID est largement adoptée dans la gestion des stocks et la logistique. Cependant, les données RFID posent de nombreux défis pour l'analyse des données : (i) les données RFID sont intrinsèquement bruitées et redondantes ; (ii) Les données RFID sont temporelles, en continu, volumineuses et doivent être traitées à la volée. En explorant la sémantique de la RFID, y compris l'emplacement, l'agrégation et les informations temporelles, nous pouvons déduire certains événements primitifs pour suivre les objets et surveiller l'état du système. En outre, nous pouvons concevoir la logique d'application comme des événements complexes, puis détecter les événements pour réaliser des applications métier plus avancées.

Les progrès récents dans les capteurs sans fil, les technologies mobiles et le traitement en continu ont conduit au déploiement de réseaux de capteurs corporels pour le suivi en temps réel de la santé d'un individu. En général, les données sur les soins de santé proviennent de capteurs hétérogènes ayant des caractéristiques distinctes, telles que des attributs divers, des relations spatio-temporelles et des caractéristiques physiologiques. En outre, les informations sur les soins de santé comportent des problèmes de confidentialité et de sécurité.

Des statistiques agrégées sur certaines partitions ont été utilisées pour identifier les clusters et assigner les caractéristiques qui ont été observées comme des valeurs plus agrégées. Les caractéristiques assignées ont ensuite été utilisées dans la modélisation prédictive pour améliorer les performances. La grande majorité de l'analyse appartient à l'analyse descriptive ou à l'analyse prédictive. En raison de la complexité de la classification, nous avons seulement résumé les approches d'analyse des données du point de vue du cycle de vie des données, en couvrant les sources de données, les caractéristiques des données et les approches, comme illustré au tableau 1.2.

Domaines d'analyse	Sources	Caractéristiques
Analyse de données structurées	Enregistrement de transaction client. Données d'expériences scientifiques.	Enregistrements structurés. Moins de volumes et temps réel.
Analyse du texte	Les journaux Email Documents d'entreprise. Règles et règlements du gouvernement. Contenu textuel des pages Web. Les réactions des citoyens et commentaires.	Non structuré. Textuelle riche. Contexte. Sémantique. Language dépendante.
Analyse du web	Diverses pages web	Intégration de texte et hyperlien. Symbolique. Métadonnées.
Analyse multimédia	Multimédia produit par la société. Multimédia généré par l'utilisateur. Surveillance. Santé et média patient.	Image, audio, vidéo. Massif. Redondance. Écart sémantique.
Analyse de réseau	Bibliométrique. Réseau de sociologie. Réseaux sociaux.	Contenu riche. Relation sociale. Bruyant et redondant. Évolution rapide.
Analyse de mobile	Applications mobile. Capteurs. RFID.	Basé sur l'emplacement. Spécifique à la personne. Informations fragmentées.

TABLE 1.2 – *Taxonomie de l'analyse Big Data [Hu et al., 2014].*

1.7 CONCLUSION

Les Big Data sont en train de devenir une nouvelle technologie, tant dans la science que dans l'industrie, avec le besoin d'apporter des mécanismes avancés d'acquisition, de gestion et d'analyse des données.

Le but de ce chapitre était de donner une idée complète sur le Big Data. Nous avons présenté diverses définitions de Big Data, son évolution et ses caractéristiques comme la variété, la vélocité et le volume. Nous avons focalisé principalement sur les composants d'architecture et les modèles opérationnels qui constituent l'écosystème de Big Data et cela a travers son cycle de vie qui regroupe quatre phases : la génération de données, l'acquisition de données, le stockage de données et l'analyse de données.

Aussi, nous avons vu des applications typiques d'analyse de données Bigdata.

Dans le chapitre suivant, nous allons nous intéresser à Hadoop qui est le framework le plus utilisé pour le traitement distribué des Big Data.

2 Hadoop : Un Framework pour le traitement des données volumineuses et distribuées

2.1 Introduction

Hadoop a attiré beaucoup d'attention de la part de l'industrie et des chercheurs. Apache Hadoop [hadoop] est une implémentation open source du framework de traitement parallèle MapReduce de Google. Il cache les détails du traitement parallèle, comprenant la distribution des données aux noeuds de traitement, le redémarrage des sous-tâches échouées et la consolidation des résultats après le calcul. Ce Framework permet aux développeurs d'écrire des programmes de traitement parallèles qui se concentrent sur leur problème de calcul plutôt que sur des problèmes de parallélisassions.

Dans ce chapitre, nous décrivons l'architecture de base de Hadoop, son fonctionnement, ses caractéristiques, ainsi que ses problèmes et défis.

2.2 Architecture

L'architecture d'Hadoop comprend deux composants principaux :

1. Un Système de fichiers distribués d'Hadoop (HDFS) [Assuncao et al., 2014] [Philip, 2011] [Castillo, 2005] : C'est un système de fichiers distribué qui stocke une grande quantité de données avec un accès haut débit aux données sur les clusters.

2. Hadoop MapReduce : C'est un Framework pour le traitement distribué des données sur les clusters.

2.2.1 Hadoop Distributed file system (HDFS)

HDFS [hdfs] [Mario et al., 2017] est un système de stockage distribué conçu pour le stockage de fichiers distribués sur un nœud HDFS. HDFS a une architecture maître / esclave. Un cluster HDFS possède un NameNode unique, un serveur maître qui gère l'espace de nom du système de fichiers et contrôle l'accès aux fichiers par les clients. [Vohra, 2016] Les données sont divisées en blocs de 64 Mo ou 128 Mo par HDFS et il maintient trois copies de chaque bloc sur des emplacements distincts appelés le DataNode.

Le NameNode exécute des opérations d'espace de nom de système de fichiers telles que l'ouverture, la fermeture et le renommage de fichiers et de répertoires tandis que le nœud de données est responsable de la réservation des demandes de lecture et d'écriture des clients du système de fichiers. Il détermine également le mappage des blocs aux DataNode.

Les DataNode effectuent la création, la suppression et la réplication de blocs sur instruction du NameNode. HDFS donne un accès à haut débit et convient à l'application qui a de grands ensembles de données. Le HDFS est hautement tolérant aux pannes et peut être déployé sur du matériel bon marché. [Li et al., 2016]

Comme le montre la figure 2.1, l'architecture HDFS [hdfs] se compose d'un seul NameNode et de plusieurs DataNodes dans un cluster. NameNode décide de la réplication des blocs de données. Dans un HDFS typique, la taille de bloc est de 64 Mo et le facteur de réplication est de 3 (la deuxième copie sur le rack local et la troisième sur le rack distant).Système de fichiers distribués d'Hadoop (HDFS) [Assuncao et al., 2014] [Philip, 2011] [Castillo, 2005] : C'est un système de fichiers distrters.

Name Node

Le Name Node [HadoopTuto] fonctionne comme une partie centrale. Il est responsable du suivi de cluster qui ont l'arborescence de tous les fichiers dans le système de fichiers. Les données de tous ces fichiers ne sont pas stockées par le Name Node. Les processus sont interactifs avec le Name Node chaque fois qu'ils sont nécessaires pour

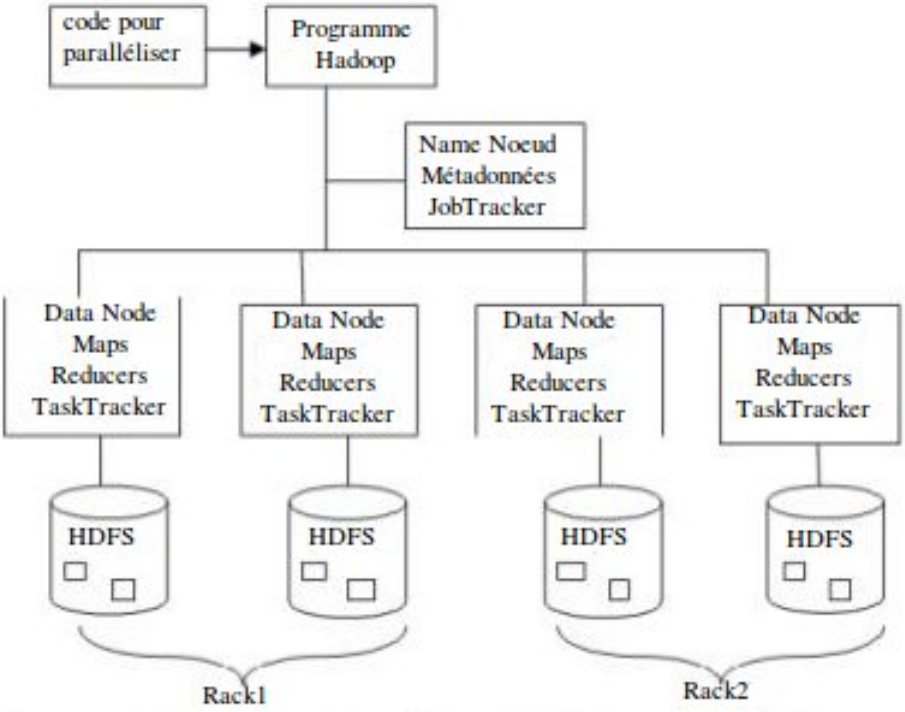

FIGURE 2.1 – *Architecture de système de fichiers distribués d'Hadoop*

localiser un fichier ou lorsqu'ils souhaitent ajouter, déplacer, supprimer ou localiser un fichier. Le noeud de nom répond aux demandes réussies en renvoyant une liste de serveurs de nœuds de données pertinents dans lesquels les données sont stockées. Il s'agit d'un seul point de défaillance dans le cluster HDFS. Le HDFS ne considère pas comme un système à haute disponibilité.

Data Node

Le Data Node [HadoopTuto], fonctionne comme esclave, il est responsable du stockage des données. Il existe plusieurs nœuds de données dans HDFS et des copies de données répliquées dans trois emplacements.

Le nœud de données et le nœud de nom fonctionnent ensemble. Le noeud de données informe le noeud de nom avec tous les blocs dont il est responsable au démarrage. Il n'y a aucun effet sur les clusters ni sur la disponibilité des données lorsque le noeud de données tombe en panne. En fait, les données sont stockées dans le nœud de données configuré avec beaucoup d'espace disque.

2.2.2 MapReduce

MapReduce [Hashem et al., 2016] [HadoopTuto] est un modèle de programme pour une technique de traitement basé sur l'informatique distribuée. Le modèle de programmation MapReduce a deux fonctions importantes, Map et Reduce.

Le fonctionnement de Map consiste à prendre les données d'entrée sous la forme d'un ensemble de paires de valeurs clés et à produire une sortie sous forme de paires de valeurs de clés intermédiaires supplémentaires. La fonction Reduce reçoit cette paire de valeurs de clé intermédiaire générée à partir de la fonction Map en tant qu'entrée et réduit ces paires de valeurs de clé intermédiaires en un ensemble de paires plus petit. La fonction Map est exécutée après la fonction Reduce selon le nom du modèle de programme MapReduce.

Le principal avantage du modèle MapReduce est évolutif pour le traitement des données sur plusieurs nœuds de calcul. Il y a trois étapes Map, Shuffle et Reduce dans le traitement de MapReduce. Le traitement de l'étape Map consiste à traiter les données d'entrée et à créer divers petits paquets de données. [Yang et al., 2016] HDFS est utilisé pour stocker les données d'entrée sous la forme d'un fichier ou d'un répertoire et ces fichiers d'entrée passent à la fonction Map pas à pas. Dans l'étape Reduce, les données de sortie reçues de la fonction Reduce proviennent de l'étape Map et sont utilisées comme données d'entrée, les traitent et produisent un nouvel ensemble de sorties stockées dans HDFS. [Bok et al., 2017].

Hadoop envoie les tâches Map et Reduce au serveur approprié dans les clusters pendant le travail MapReduce. Il incombe à Hadoop de gérer les détails des tâches pertinentes, comme la vérification de l'exécution des tâches, l'émission des tâches et la copie des données autour du cluster entre les nœuds. Diverses tâches informatiques qui réduisent le trafic réseau sont placées sur des noeuds avec des données sur des disques locaux. Lorsque les tâches sont terminées, le cluster collecte les données de sortie et les réduits à un format approprié pour les renvoyer au serveur Hadoop. [Hashem et al., 2016]

Job Tracker

Job Tracker [HadoopTuto] est un nœud. Le processus d'exécution des tâches est contrôlé par Job Tracker et coordonne tous les travaux en planifiant la tâche qui s'exé-

cute sur le système pour s'exécuter sur Task Tracker. Toutes les ressources de traitement du cluster Hadoop sont gérées par Job Tracker et Job Tracker les planifie dans Task Tracker pour exécution conformément à la demande d'application.

Job Tracker a la capacité de recevoir des mises à jour de statut pour suivre la progression du processus à partir de Task Tracker pendant l'exécution du processus. Ses coordonnées pour gérer des échecs si nécessaire. Il est nécessaire d'exécuter Job Tracker sur le nœud maître, car il est responsable de la coordination de tous les travaux MapReduce du cluster.

Task Tracker

Task Tracker [HadoopTuto] est également un nœud qui reçoit des tâches de Jobs Tracker. Il est du devoir du Task Tracker d'exécuter la tâche et si n'importe quelle panne s'est produite alors renvoyez le rapport au Job Tracker au sujet de cet échec.

Job Tracker coordonne et aide à résoudre les problèmes qui surviennent lors du traitement des tâches. Les ressources de traitement sur chaque nœud esclave sont gérées par Task Tracker sous la forme de slots. [Geetha et al., 2016] le nombre de slots pour les tâches map-reduce sont fixes. Le nombre de tâches map-reduce peut être exécuté à la fois sur le nœud esclave indique le nombre total de slots map-reduce.

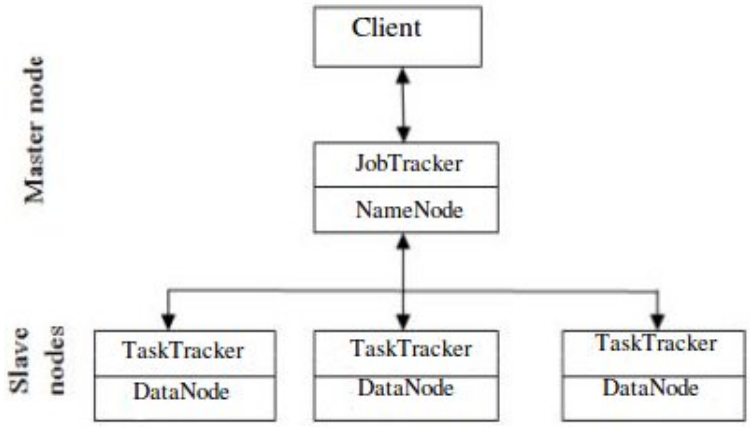

FIGURE 2.2 – *Architecture MapReduce d'Hadoop*

2.3 Fonctionnement

Lorsqu'un utilisateur soumet un travail à Hadoop. Il doit spécifier l'emplacement des fichiers d'entrée et de sortie dans HDFS. La fonction Map and Reduce est implémentée dans les classes Java sous la forme de fichiers jar et les différents paramètres doivent être corrigés afin de permettre l'exécution des tâches pour la configuration des tâches. Ensuite, l'utilisateur de Hadoop soumet un travail la configuration du travail au Job Tracker. Maintenant Job Tracker distribue le travail et les configurations aux tâches de planification. [Seay et al., 2015].

Il contrôle également les travaux et fournit des informations sur l'état et le diagnostic à l'utilisateur Hadoop. La responsabilité de Task Tracker est d'exécuter une tâche sur un nœud différent selon l'implémentation de MapReduce et de sauvegarder la sortie dans les fichiers de sortie sur HDFS.

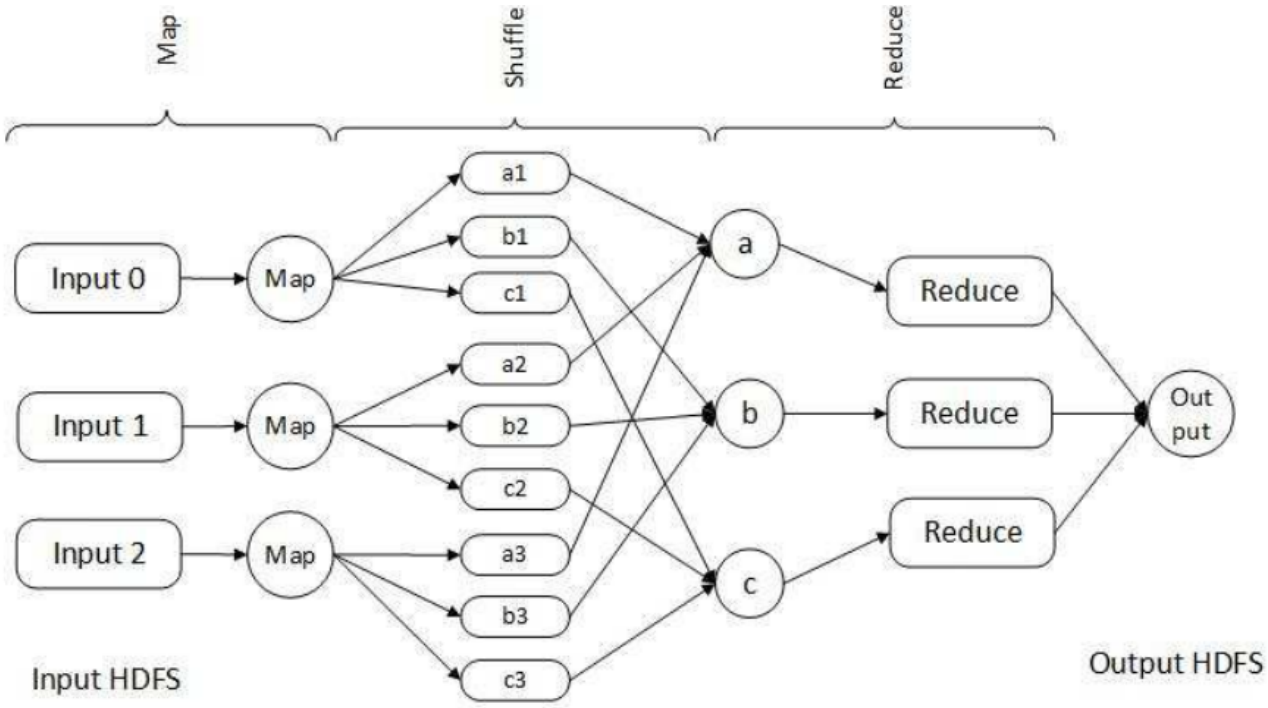

Figure 2.3 – *Processus MapReduce d'Hadoop*

La figure 2.3. définie la méthode traditionnelle de MapReduce, mais maintenant Hadoop travaille avec un nouveau paramètre nommé YARN qui donne une approche vraiment nouvelle au traitement des données, ce qui donne un moyen facile de traiter les données avec une seule plate-forme et fournit une nouvelle perspective analytique.

2.3.1 Hadoop YARN

Dans la nouvelle génération de framework Hadoop, YARN (Yet Another Resource Négociateur) [Bhavin J. et Vinodkumar L., 2015] est une partie architecturale de base (Figure 2.4) qui permet le traitement de données multiples comme le traitement par lots, la science des données, le streaming temps réel et le SQL interactif. Elle permet aussi de gérer le traitement des données avec une plateforme unique et fournir une nouvelle perspective à l'analyse.

Le but fondamental de YARN [Lin et al., 2016] est de diviser les fonctionnalités de Job Tracker / Task Tracker en démons séparés. Dans cette nouvelle version de Hadoop, un gestionnaire de ressources global, un gestionnaire de nœud esclave pour chaque nœud, maître d'application et conteneur pour chaque application (un seul travail ou peut-être un DAG de travaux) s'exécute sur un gestionnaire de nœud [Sarah, 2016]

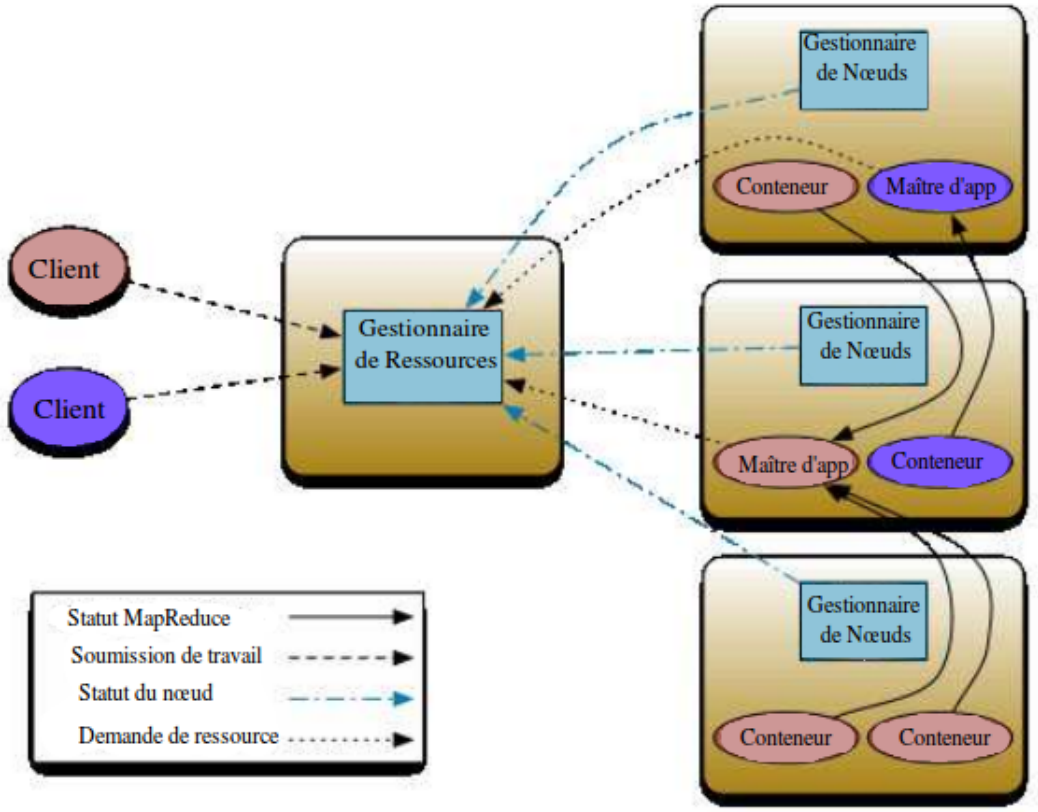

FIGURE 2.4 – *Architecture YARN*

Le Gestionnaire de Ressources et le Gestionnaire de Nœuds [Thomas et Qiuyi, 2016] fonctionnent ensemble comme un framework traitement-données. Un gestionnaire de ressources global s'exécute en tant que démon maître qui arbitre les ressources de cluster parmi les applications concurrentes disponibles dans le système.

Le travail de gestionnaire de ressources consiste à suivre les nœuds s'exécutant sur le cluster et les ressources disponibles. Il prend également en considération les demandes soumises par l'utilisateur devrait obtenir des ressources chaque fois que nécessaire. Il est responsable d'allouer ces ressources à l'application utilisateur en fonction d'une priorité d'application, d'une capacité de file d'attente, de la localité de données, etc.

Dans Hadoop 2.0 Task Tracker est remplacé par Node Manager qui est une version plus générique et efficace. Le gestionnaire de noeud n'a pas de conteneur de ressources créé dynamiquement ou slots Map et Reduce fixe comme dans Task Tracker. La taille du conteneur dépend du nombre de ressources qu'il contient comme la mémoire, le disque, CPU, réseau, etc. Le gestionnaire de nœuds est responsable de l'application du conteneur et surveille l'utilisation de ses ressources comme le CPU, le disque, la mémoire et le réseau.

Dans Hadoop 2.0 Job Tracker est légèrement modifié avec un peu de perspicacité de conception et nous utilisons la terminologie pour cela en tant que Application Master. Le maître d'application a la capacité de n'importe quel type de tâche à l'intérieur du conteneur. Ainsi, lorsqu'un utilisateur soumet une application, Le maître d'application est chargé de coordonner l'exécution de toutes les tâches dans l'application. Le maître d'application a une plus grande responsabilité qui était précédemment assignée à Job Tracker pour tous les travaux comme la tâche de surveillance, le démarrage des tâches échouées, le calcul des valeurs totales du compteur d'applications et l'exécution spéculative de tâches lentes. Le gestionnaire de nœuds contrôlait le maître d'application et les tâches qui appartenaient à l'application s'exécutant dans le conteneur de ressources.

Dans YARN le rôle de l'application distribuée est dégradé par MapReduce mais il est toujours très populaire et utile et maintenant il est connu comme MapReduce version 2 (MR2). MR2 est une implémentation classique du moteur MapReduce qui s'exécute en haut de YARN. MapReduce dans Hadoop 2 maintient la compatibilité de l'API avec la version stable précédente alors que tous les travaux de MapReduce fonctionnent toujours avec YARN via une recompilation.

2.4 Caractéristiques

Hadoop présente de nombreux avantages, et les fonctionnalités suivantes rendent Hadoop particulièrement adapté à la gestion et à l'analyse de Big Data :

— **Évolutivité :** Hadoop permet d'augmenter et de réduire l'infrastructure matérielle sans avoir à modifier les formats de données. Le système redistribuera automatiquement les données et les tâches de calcul pour s'adapter aux changements de matériel.

— **Rentabilité :** Hadoop apporte un calcul massivement parallèle aux serveurs de base, conduisant à une réduction importante du coût par téraoctet de stockage, ce qui rend le calcul massivement parallèle abordable pour le volume croissant de big data.

— **Flexibilité :** Hadoop est libre de tout schéma et capable d'absorber n'importe quel type de données provenant d'un nombre quelconque de sources. En outre, différents types de données provenant de sources multiples peuvent être agrégés dans Hadoop pour une analyse plus approfondie. Ainsi, de nombreux défis de Big Data peuvent être résolus.

— **Tolérance aux pannes :** les données manquantes et les échecs de calcul sont courants dans les analyses de Big Data. Hadoop peut récupérer les données et les échecs de calcul causés par une panne de nœud ou une congestion du réseau.

2.5 Problèmes et défis

Les ordonnanceurs Hadoop sont conçus pour une meilleure utilisation des ressources et une amélioration des performances. [Suresh et Gopalan, 2014] Les performances de l'ordonnanceur ont été affectées par certains problèmes et des efforts de recherche sont encore nécessaires pour améliorer l'efficacité des ordonnanceurs Hadoop.

Certains des principaux défis de recherche sont discutés ici comme Efficacité énergétique, Schéma de Mapping, Équilibrage de charge, Automatisation et configuration, optimisation de la réorganisation des données, Optimisation des performances, Équité, Localité des données et Synchronisation.

Efficacité énergétique

La croissance continue de la taille des données dans un data center contenant un cluster Hadoop pour prendre en charge de nombreux utilisateurs et opérations d'un grand nombre de données doivent être effectuées dans ce data center. [Zhai et al., 2016]

Pour surmonter ces opérations, une grande quantité d'énergie requise dans le centre de données et augmenter l'énergie dans le data center augmentera le coût global. La minimisation de l'énergie dans le centre de données est un grand défi pour Hadoop.

Equilibrage de charge

Il y a deux étapes Map et Reduce liés avec l'étape de partition. La partition est considérée comme le facteur principal pour mesurer les performances.

Par défaut, les données sont également réparties par les algorithmes de partition et gèrent le déséquilibre du système en cas de données asymétriques. Parce que seule la clé est considérée dans le traitement pas la taille des données. Donc, le problème d'équilibrage de charge [Dongchul et Yang-Suk, 2015] se produit dans cette situation.

Schéma de mapping

Il est nécessaire d'affecter des ensembles d'entrée au réducteur afin que le réducteur reçoive toutes les entrées possibles avant l'exécution pour chaque sortie. Aucune entrée fixe ne peut être assignée parce que le réducteur a une capacité limitée d'entrées dues tandis que la taille d'entrée individuelle peut très bien conduire à augmenter le coût de communication.

De nombreuses solutions ont été développées pour faire face à la restriction de la taille des entrées, mais l'optimisation des coûts de communication entre les phases map et reduce a rarement été observée. Ainsi, un schéma map [PARK et al., 2016] qui pourrait être utile pour minimiser les coûts de communication est requis.

Automatisation et configuration

L'automatisation et la configuration [Shi et al., 2014] aident au déploiement du cluster Hadoop en définissant tous les paramètres. Pour une configuration correcte, le matériel et la quantité de travail sont connus au moment du déploiement.

Lors de la configuration, une petite erreur peut être une raison pour l'exécution inefficace du travail entraîne une dégradation des performances. Pour surmonter ces types de problèmes, nous devons développer une nouvelle technique et des algorithmes qui effectuent le calcul de telle manière que l'établissement puisse être effectué de manière efficace.

Combinaison de données optimisées

Dans MapReduce, les performances globales du système diminuent car l'entrée / sortie intensive du disque augmente le temps d'exécution pendant la phase de combinaison. [Li et al., 2016] Donc, réduire le temps d'exécution pendant la phase de combinaison est plus difficile.

Optimisation des performances

Il y a plusieurs facteurs tels que l'ordonnancement, le temps d'initialisation des tâches et la dégradation de la performance de la surveillance dans divers travaux, mais ils ne prennent pas en charge les fonctions telles que le chevauchement des phases MapReduce.

Pour améliorer Map and Reduce, certains aspects nécessitent une optimisation supplémentaire, comme la création d'index, l'exécution rapide de requêtes et la réutilisation des résultats calculés précédemment.

Équité

Équité se référer à l'équité des algorithmes d'ordonnancement sur la répartition équitable des algorithmes divisant les ressources entre les utilisateurs. Une mesure équitable des ressources est nécessaire parmi les utilisateurs sans famine. Le cluster entier est utilisé par map-reduce avec une lourde charge de travail car ce travail léger n'obtient pas le temps de réponse souhaité.

L'équité [Zhigang Wang, 2015] traite de la dépendance et de la localité entre les phases

Map et Reduce. S'il y a une part égale de travail de mappage et de réduction et que la distribution des fichiers d'entrée a été effectuée en clusters, la dégradation des performances du temps de réponse et du débit doit être effectuée.

Localité de données

La distance entre le noeud de tâche et le noeud d'entrée est appelée localité. Le taux de transfert de données dépend de la localité. Le temps de transfert de données sera court si le noeud de calcul est proche du noeud d'entrée.

La plupart du temps, il n'est pas possible d'acquérir la localité de nœud, dans une telle situation, nous acquérons la localité de rack en effectuant la tâche au même rack. Pour prendre en charge les périphériques matériels, un cadre métallique est utilisé, appelé Rack.

Synchronisation

La synchronisation [Jia et al., 2014] est un facteur de signification dans MapReduce. Le processus de transfert des données intermédiaires de sortie du processus de mappage en tant que données d'entrée pour le processus reduce est connu sous le nom de synchronisation. Tandis que le processus reduce est démarré après l'achèvement du processus Map, si un processus Map est retardé, les performances de l'ensemble du processus vont ralentir.

C'est un problème commun qui émerge généralement dans un environnement hétérogène car chaque nœud a une bande passante, un matériel et une infrastructure de calcul uniques, ce qui conduit à une réduction des performances du cluster Hadoop.

2.6 Conclusion

Nous avons présenté dans ce chapitre les différents composants d'Hadoop, principalement, HDFS et le modèle de programmation MapReduce. Nous avons étalé sur les aspects théoriques pour aborder les approches et techniques pour l'ordonnancement des tâches dans le chapitre suivant.

3 Approches et techniques pour l'ordonnancement : Etat de l'art

3.1 Introduction

Dans les environnements à forte intensité en données comme Hadoop, le volume de données impliquées est énorme. Le transfert de données entre des nœuds de calcul prend beaucoup de temps en fonction de la taille des données et de la capacité du réseau entre les nœuds participants. Par conséquent, la plupart des techniques d'ordonnancement visent à optimiser les transferts de données en exploitant la localité des données.

L'objectif principal de ce chapitre est de présenter une classification des ordonnanceurs des tâches dans l'environnement Hadoop. De plus, il fait une étude comparative entre divers algorithmes d'ordonnancement de tâches.

3.2 L'ordonnancement dans Hadoop :

3.2.1 Définition :

L'ordonnancement est connu pour être un problème multidimensionnel car il doit optimiser un ensemble de métriques sous les contraintes de la nature des travaux et des ressources limitées [Brucker, 2001] [Lawler et al., 1993] .

L'ordonnancement dans MapReduce ne fait pas exception. Un algorithme d'ordonnancement MapReduce doit : (i) planifier un ensemble d'entités telles que les utilisateurs, les travaux et les tâches ; (ii) s'adapter aux données, aux ressources et aux caractéristiques de la charge de travail de l'environnement d'exécution ; et (iii) satisfaire à un

ensemble d'exigences de qualité telles que la localisation optimale des données, temps de réponse et le respect des délais, l'utilisation optimale des ressources, l'équité de l'allocation, l'efficacité énergétique globale, etc.

Dans ce chapitre, nous allons étudier les algorithmes d'ordonnancements selon une classification hiérarchique.

3.3 Les Exigences de Qualité d'ordonnancement

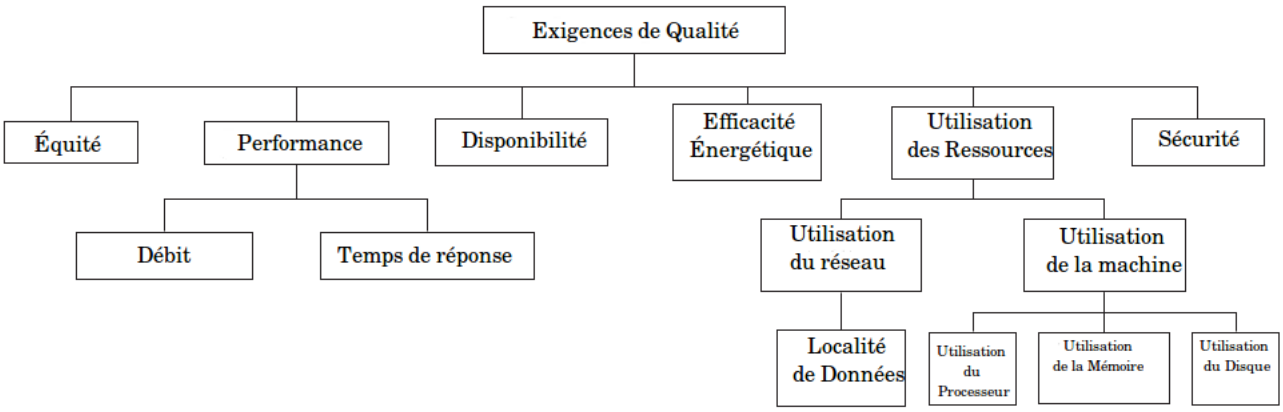

Figure 3.1 – *Classification basée sur les exigences de qualité [Tiwari et al., 2015]*

Nous définissons les attributs de qualité les plus communs (primaires ou dérivés) pris en compte par les algorithmes d'ordonnancement MapReduce.

Équité :

L'équité signifie donner à chaque travail dans le système une possibilité égale ou justifiée d'exécution. Les big data d'entreprise sont de plus en plus utilisés pour effectuer divers travaux d'analyse de données tels que la détermination de la demande, la détection de la fraude et l'analyse de la performance / disponibilité par différents utilisateurs. Il est fort probable que plusieurs utilisateurs, chacun effectuant une analyse de données spécifique, soient soumis par les utilisateurs au cluster en même temps.

Pour garantir l'équité vis-à-vis des utilisateurs et des tâches, les ressources doivent être réparties entre elles, soit de manière égale, soit en fonction de leurs priorités, de leur taille ou de la rapidité d'exécution requise pour l'exécution de leurs tâches. Les algorithmes d'ordonnancement sont les seuls moyens d'y parvenir. C'est l'une des importantes exigences de qualité que plusieurs algorithmes d'ordonnancement de

MapReduce ont essayé de satisfaire. Dans la plupart des cas, une approche commune utilisée pour assurer l'équité consiste à planifier les tâches de telle sorte que tous les utilisateurs / travaux obtiennent un pourcentage égal de slots.

Débit :

Dans le contexte d'un système MapReduce, le débit implique le nombre de travaux / tâches accomplies sur un intervalle de temps donné. Un débit élevé peut être atteint en améliorant d'autres mesures de qualité telles que la localisation des données, l'efficacité énergétique et l'utilisation efficace des ressources, parfois, une trop grande importance accordée à la localisation des données peut en fait réduire le débit du système.

Temps de réponse :

Le temps de réponse signifie qu'un travail MapReduce doit respecter un temps de réponse convenu d'un travail pour la plupart des exécutions de ce travail. Le nombre de travaux pour lesquels le temps de réponse ne respecte pas l'accord devrait se situer dans une limite de tolérance convenue exprimée en pourcentage du nombre total des travaux soumis. Bien que, traditionnellement, MapReduce soit un système de traitement par lots, il est de plus en plus utilisé pour des travaux interactifs comme les requêtes SQL sur de grands ensembles de données. Dans ce contexte, le temps de réponse devient une exigence clé pour les jobs MapReduce.

Disponibilité :

La disponibilité est définie comme le pourcentage de temps pendant lequel un système est opérationnel et répond aux demandes. Il peut également être défini comme le délai moyen entre les échecs de réponse aux demandes des utilisateurs. Si un système MapReduce s'exécute sur des machines distribuées, sur une infrastructure de cloud appartenant à différentes personnes ou utilise du matériel de base, la disponibilité devient une préoccupation importante pour toutes les parties prenantes. L'ordennanceur MapReduce doit ensuite s'assurer que tous les travaux sont terminés, même dans des environnements aussi volatiles.

Efficacité énergétique :

L'exigence d'efficacité énergétique d'un système MapReduce exige que les clusters consomment le moins d'énergie possible lors de l'exécution d'une charge de travail donnée, sans affecter les performances. L'augmentation du volume de données et l'utilisation de serveurs de base dans les clusters MapReduce ont généré des milliers de nœuds dans les clusters d'entreprise.

De telles clusters consomment d'énormes quantités d'énergie et augmentent le coût opérationnel des centres de données. Par conséquent, l'augmentation de l'efficacité énergétique des clusters MapReduce a attiré beaucoup d'attention. Les faibles niveaux d'utilisation des serveurs, dus à la faible charge et à la duplication des données, entraînent également une faible efficacité énergétique dans les clusters MapReduce. des algorithmes d'ordonnancement [Shi et Srivastava, 2010] ont été proposés pour améliorer l'efficacité énergétique de MapReduce ;

Localité de données :

Contrairement à d'autres attributs de qualité, il s'agit d'une exigence de qualité spéciale pour un système MapReduce car l'un des principes du modèle de programmation MapReduce consiste à rapprocher l'exécution des tâches près des données pour l'exécution des données locales. Cette exigence exige que l'algorithme d'ordonnancement doit s'efforcer d'augmenter les instances des exécutions de processus de données locales ou de rack.

Différents algorithmes d'ordonnancement utilisent différentes heuristiques et méthodes mathématiques pour ordonnancer différentes entités, telles que le travail, la tâche map ou la tâche reduce, afin d'obtenir un nombre élevé de processus exécutés sur des données locales. L'augmentation des exécutions de tâches locales de données aide également à minimiser la contention du réseau (c'est-à-dire l'utilisation efficace des ressources) et entraîne un débit plus élevé.

Utilisation des ressources :

L'utilisation efficace des ressources implique la minimisation du temps d'arrêt de l'unité centrale, des E / S et des ressources réseau, ou des états d'inactivité ou de contention. Dans un environnement partagé étendu, l'utilisation efficace de ressources

telles que le processeur, la mémoire et les E / S devient essentielle pour assurer la meilleure utilisation possible des ressources disponibles par le plus grand nombre possible d'utilisateurs / travaux. Assurer un équilibre global dans l'utilisation des différents types de ressources à travers le cluster est une partie importante de l'objectif d'utilisation des ressources. Sans cet équilibre, le cluster serait faussé dans l'utilisation des ressources et, finalement, montrerait un impact négatif sur la performance du travail.

Par exemple, un travail MapReduce peut nécessiter beaucoup d'UC et de mémoire, tandis qu'un autre est plus gourmand en E / S. Si l'ordennanceureur démarre deux travaux gourmandes en UC et qu'il commence à partager la même infrastructure physique, la ressource d'E / S peut rester sous-utilisée alors qu'il y a un conflit pour le UC. Ce biais dans l'utilisation des ressources augmentera le temps d'achèvement des deux travaux. Si deux travaux différents utilisent le même jeu de données, se concentrer uniquement sur l'amélioration de la localité de données peut entraîner des conflits pour quelques ressources et peut augmenter l'asymétrie d'utilisation. Par conséquent, l'ordennanceur doit trouver le bon équilibre afin que l'amélioration de la localisation des données n'introduise pas d'asymétrie dans le cluster.

Sécurité :

La sécurité d'un cluster MapReduce peut être définie comme ayant un accès hautement contrôlé aux données, aux nœuds et aux processus s'exécutant sur tous les nœuds afin d'éviter tout risque d'utilisation abusive et de corruption.

L'ordennanceur MapReduce s'appuie actuellement sur la liste de contrôle d'accès (ACL) du système Unix sous-jacent pour la sécurité d'accès aux processus. Il utilise un système de fichiers distribué Kerberos pour la sécurité d'accès aux données. Les jetons de la délégation ont été introduits pour assurer la mise à l'échelle de Kerberos dans des grands clusters. Cependant, nous n'avons trouvé aucun algorithme d'ordennancement conçu pour assurer la sécurité des données, des nœuds ou des processus sur différents nœuds.

3.4 Classification des ordonnanceurs dans Hadoop

Dans un environnement partagé, l'ordonnanceur sélectionne d'abord un utilisateur dont le travail doit être planifié. Un utilisateur peut avoir une application basée sur un workflow composée de plusieurs tâches de graphe directement acyclique (DAG) ou de nombreux travaux individuels à exécuter. Les tâches d'une application basée sur le workflow sont planifiées en fonction de leur contrainte de précédence. Les travaux indépendants sont sélectionnés pour l'ordonnancement en fonction des préférences ou des stratégies de l'utilisateur. Une fois le travail sélectionné, ses tâches Map, Reduce ou spéculatives sont ordonnancées.

Différents algorithmes décisionnels peuvent être utilisés à chaque niveau en fonction de l'objectif à atteindre. Dans [95], une classification hiérarchique de plusieurs niveaux est proposée. Cette classification est représentée à la figure 3.2.

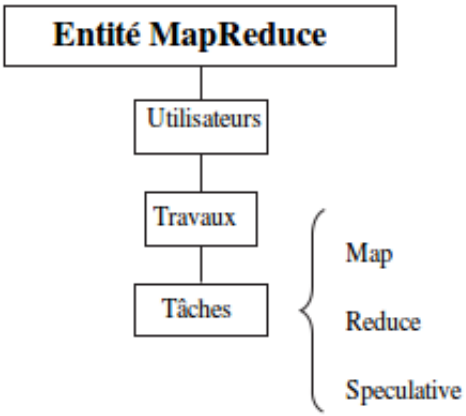

Figure 3.2 – *Les entités MapReduce hiérarchiques pour être ordonnancé*

1. **Niveau de l'utilisateur :** L'algorithme d'ordonnancement au niveau de l'utilisateur sélectionne un utilisateur dans la file d'attente d'utilisateur pour planifier ses tâches dans le cluster.
2. **Niveau du travail :** L'algorithme d'ordonnancement au niveau du travail sélectionne un travail dans la file d'attente pour planifier ses tâches. Un cluster peut avoir une seule file d'attente de travail, dans laquelle les travaux de tous les utilisateurs sont soumis, ou plusieurs files d'attente de travaux, chacune contenant des travaux d'utilisateurs différents.
3. **Niveau tâche :** Le troisième niveau de l'ordonnancement hiérarchique implique la sélection d'une tâche, de l'utilisateur sélectionné et du travail pour l'affectation

sur le nœud disponible. Le modèle de programmation MapReduce crée deux types de tâches Map et Reduce qui doivent être l'ordonnancer dans les slots respectifs. Un autre type de tâche créé au cours de l'exécution de MapReduce est la tâche spéculative pour gérer les tâches map / reduce retardataires. Ceux-ci sont ordennancer dans Map et Reduce, en fonction du type de tâches qu'ils dupliquent.

(a) **Niveau de la tâche Map :** Les tâches map exécutent la première phase du programme MapReduce en analysant l'ensemble de données d'entrée. Un ensemble de tâches map, chacune traitant un bloc de données, est programmé pour s'exécuter simultanément sur un ensemble de noeuds disponibles. L'algorithme d'ordonnancement doit optimiser plusieurs attributs de qualité associés aux travaux MapReduce et au cluster lors de l'ordonnancement de ces tâches map. Pour cette raison, un grand nombre d'algorithmes d'ordonnancement ont été proposés pour améliorer l'ordonnancement des tâches map.

(b) **Niveau de la tâche Reduce :** Une fois qu'une tâche Map d'un travail est terminée, MapReduce commence l'ordonnancement les tâches reduce afin que l'exécution des tâches mapet le réarrangement des données intermédiaires puissent s'exécuter en parallèle et que le travail nécessite moins de temps d'exécution. Ceci est connu comme "early-shuffle". MapReduce sélectionne aléatoirement une tâche reduce du travail sélectionné pour le slot reduce disponible.

(c) **Niveau de tâche spéculatif :** Dans Hadoop, les tâches spéculatives sont identifiées en fonction d'une métrique appelée Score de progression des différentes tâches d'un travail. Cette métrique est calculée à la fois au niveau de la tâche map et au niveau de la tâche reduce.

Pour une tâche map, cette mesure indique la fraction de l'entrée lue jusqu'à présent. Pour une tâche reduce, d'abord son exécution est divisée en trois phases logiques, puis la métrique est calculée pour chaque phase, et finalement, les valeurs de phase sont additionnées pour obtenir le score global pour toute la tâche reduce. Pour marquer une tâche map ou reduce comme traînard, Hadoop considère un seuil basé sur la valeur moyenne du score de progression et le temps passé par la tâche en cours d'exécution. Lorsque

ces conditions sont remplies, Hadoop marque la tâche en tant que traînard. Si une tâche devient un traînard, une copie de la tâche est "exécutée de manière spéculative" dans le but d'améliorer la fiabilité et le temps de réponse. Dans Hadoop, l'exécution spéculative est basée sur plusieurs hypothèses : le cluster est homogène, toutes les tâches progressent au même rythme et toutes les tâches reduce traitent la même quantité de données . Cependant, ces hypothèses ne sont pas toujours vraies car le cluster peut être hétérogène, le taux de progression de différentes tâches peut être différent (certaines peuvent être locales, d'autres distants, d'autres plus intensives) et différentes tâches reduce peuvent traiter différentes quantités de données en fonction des clés intermédiaires et de leurs volumes de données. Ensuite, l'ordonnancement de tâches spéculatives a la possibilité d'augmenter le temps d'achèvement de tous les travaux, entraînant ainsi un gaspillage de ressources.Pour que les exécutions de tâches spéculatives soient efficaces, l'ordonnanceur doit considérer les aspects suivants : identifier une tâche traînard le plus tôt possible afin que sa tâche spéculative soit planifiée rapidement, reconnaître la concurrence pour les ressources (réseau, CPU et autres) causées par la spéculation (rien que de dupliquer) des tâches, et identifier les nœuds.

Pour que les exécutions de tâches spéculatives soient efficaces, l'ordonnanceur doit considérer les aspects suivants : identifier une tâche traînard le plus tôt possible afin que sa tâche spéculative soit planifiée rapidement, reconnaître la concurrence pour les ressources (réseau, CPU et autres) causées par la spéculation (rien que de dupliquer) des tâches, et identifier les nœudsabord son exécution est divisée en trois phases logiques, puis la métrique est calculée pour chaque phase, et finalement, les valeurs de phase sont additionnées pour obtenir le score global pour toute la tâche reduce. Pour marquer une tâche map ou reduce comme traînard, Hadoop considère un seuil basé sur la valeur moyenne du score de progression et le temps passé par la tâche en cours d'exécution. Lorsque ces conditions sont remplies, Hadoop marque la tâche en tant que traînard. Si une tâche devient un traînard, une copie de la tâche est "exécutée de manière spéculative" dans le but d'améliorer la fiabilité et le temps de réponse.

Dans Hadoop, l'exécution spéculative est basée sur plusieurs hypothèses : le cluster est homogène, toutes les tâches progressent au même rythme et toutes les tâches reduce traitent la même quantité de données [Zaharia et al., 2008]. Cependant, ces hypothèses ne sont pas toujours vraies car le cluster peut être hétérogène, le taux de progression de différentes tâches peut être différent (certaines peuvent être locales, d'autres distants, d'autres plus intensives) et différentes tâches reduce peuvent traiter différentes quantités de données en fonction des clés intermédiaires et de leurs volumes de données. Ensuite, l'ordonnancement de tâches spéculatives a la possibilité d'augmenter le temps d'achèvement de tous les travaux, entraînant ainsi un gaspillage de ressources.

Pour que les exécutions de tâches spéculatives soient efficaces, l'ordonnanceur doit considérer les aspects suivants : identifier une tâche traînard le plus tôt possible afin que sa tâche spéculative soit planifiée rapidement, reconnaître la concurrence pour les ressources (réseau, CPU et autres) causées par la spéculation (rien que de dupliquer) des tâches, et identifier les nœuds.

Dans ce qui suit, nous allons étudier les travaux liés aux algorithmes d'ordonnancement qui exploitent la localité de données.

3.4.1 Ordonnancement des travaux

Classification des ordonnanceurs utilisés dans le Big Data

Les ordonnanceurs des travaux [Jiong Xie, 2013] [Suresh et Gopalan, 2014] peuvons être classés en termes d'aspect : environnement, priorité, ressources (temps CPU, slot libre, espace disque, utilisation des E / S), temps et stratégies. L'idée principale derrière l'ordonnancement est de minimiser les frais généraux, les ressources, le temps de réalisation et d'optimiser le débit en allouant les travaux aux nœuds. Dans cette section, la classification des ordonnanceurs se fait en fonction des stratégies d'ordonnancement, du temps et des ressources [Usama et al., 2017].

Nous allons maintenant voir les différents ordonnanceurs de travaux présents dans Hadoop :

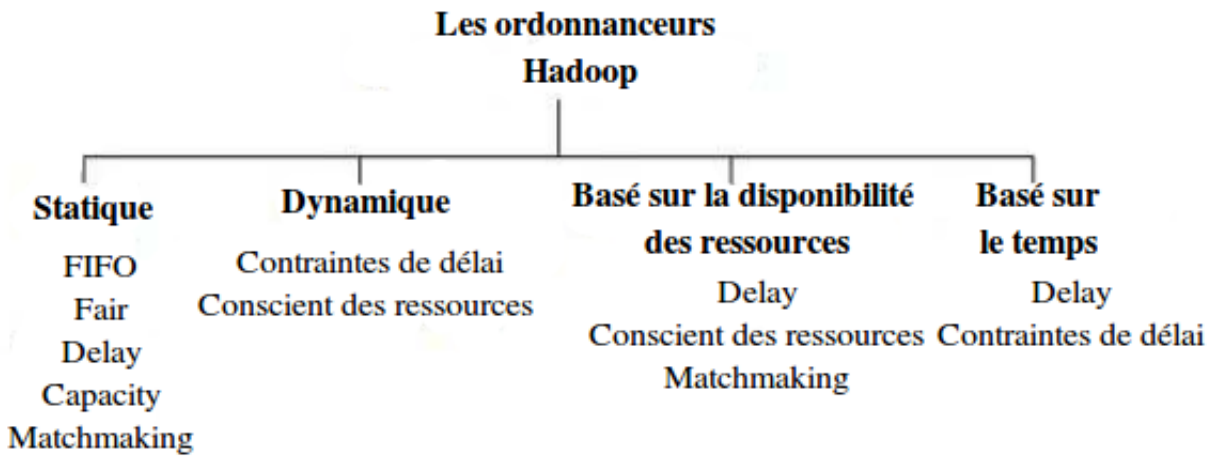

FIGURE 3.3 – *Classification des ordonnanceurs des travaux hadoop [Usama et al., 2017]*

1. **Les ordonnanceurs statiques :** Les stratégies d'ordonnancement statiques allouent un travail au noeud avant que l'exécution du programme ne commence. Le nœud de traitement et le temps d'exécution du travail ne sont connus qu'au moment de la compilation. L'objectif principal de ce type d'ordonnanceurs est de minimiser le temps de traitement global des travaux en cours d'exécution. La stratégie FIFO (First In First Out), Delay, Capacity, LATE (Temps d'approximation le plus long pour finir) et Matchmaking.

2. **Les ordonnanceurs dynamiques :** Les stratégies d'ordonnancement dynamique [Shao et al., 2016] allouent le travail au noeud au moment de l'exécution. Les ordonnanceurs ont peu de connaissance sur la ressource avant l'exécution, mais l'environnement dans lequel le travail s'exécutera est totalement inconnu. Lorsque le travail commence l'exécution dans un environnement dynamique, les décisions sont prises et appliquées au noeud. Resource Aware et Deadline Constrain font partie de la catégorie des ordonnanceurs dynamiques.

3. **Les ordonnanceurs Basés sur la disponibilité des ressources :** Fondamentalement, cette stratégie d'ordonnancement est basée sur les besoins en ressources du travail. Sous cette stratégie, l'utilisation des ressources (comme les E / S, l'utilisation de la mémoire, le stockage sur disque et le temps CPU) et les performances du travail sont améliorées. L'ordonnanceur Delay, l'ordonnanceur matchmaking et les ordonnanceurs Conscient des ressources sont basés sur la

disponibilité des ressources.

4. **Les ordonnanceurs basés sur le temps :** Cette stratégie d'ordonnancement est basée sur le temps, ici l'achèvement du travail dépend du délai fourni par l'utilisateur. Dans cette stratégie d'ordonnancement, il y a un délai dans lequel le travail doit se terminé. Deux stratégies d'ordonnancement, Contrainte Delai et Delay, sont utilisées lorsque nous affectons un travail en fonction du temps.

Description des ordonnanceurs utilisés dans hadoop

1. **Premier arrivé Premier sorti (FIFO)** La stratégie d'ordonnancement FIFO [Yoo et Sim, 2011] est la stratégie par défaut utilisée dans Hadoop. Elle donne préférence au premier venu des travaux. Lorsque de nouveaux travaux arrivent Job Tracker tire le premier travail en premier de la file d'attente. [Divya S, 2015] Ici, peu importe la taille du travail ou n'importe quelle priorité, seuls les travaux suivants sont autorisés dans la file d'attente et les travaux en attente attendent jusqu'à ce que le premier travail soit exécuté. La stratégie d'ordonnancement FIFO est utilisée lorsque l'ordre d'exécution des travaux n'a pas d'importance.

   Avantages

   — La technique d'ordonnancement FIFO est simple et efficace parmi tous les ordonnanceurs.
   — Les travaux sont exécutés dans le même ordre que celui dans lequel elles ont été soumises.

   Inconvénient

   — L'inconvénient majeur de l'ordonnancement FIFO est qu'elle n'est pas préemptive. Pour cette raison, il ne convient pas pour les travaux interactifs.
   — Un autre inconvénient est qu'un long processus retardera tous les travaux derrière lui.
   — L'ordonnancement FIFO ne prend pas en compte l'équilibre de l'allocation des ressources entre les longs travaux et les courts travaux.
   — Il réduit la localité de données et la famine des travaux.

2. **L'ordonnanceur équitable (Fair)** Facebook a développé l'ordonnanceur équitable [Yoo et Sim, 2011], l'idée principale derrière l'ordonnanceur équitable est d'allouer un partage égal des ressources à chaque travail. Il crée des groupes de travail basés sur des attributs configurables tels que le nom d'utilisateur, appelés pools. il effectue l'équité dans le partage des ressources entre les pools. Les pools contrôlent également les propriétés configurables du travail et les propriétés configurables déterminent les pools dans lesquels le travail est placé.

   Chaque utilisateur a ses propres pools avec une part de partage minimum pour chacun. Une petite quantité de slot obtient toujours par un pool. Par défaut, il existe une allocation équitable des ressources entre les pools avec l'emplacement de tâche MapReduce. Si un groupe est libre, c'est-à-dire qu'il n'utilise pas ses emplacements, alors ses emplacements inactifs seront utilisés par un autre groupe. Si le même utilisateur ou le même pool envoie trop de travaux que l'ordonnanceur juste peut limiter ces travaux en marquant des travaux comme non exécutables. Si un seul travail est en cours d'exécution, il peut utiliser le cluster entier.

   Avantages

   — L'ordonnanceur effectue une réallocation de ressources équitable et dynamique.

   — Permettre une réponse rapide aux petits travaux parmi les grands travaux.

   Inconvénients

   — L'ordonnanceur a des configurations plus compliquées.

   — Il ne prend pas en compte le poids de chaque travail, ce qui conduit à des performances déséquilibrées dans chaque pool / nœud.

   — Les pools ont une limitation sur le nombre de travaux en cours d'exécution.

3. **l'ordonnanceur Capacité** À l'origine, Yahoo a développé l'ordonnanceur Capacité [Gautam et al., 2015] [Yoo et Sim, 2011]. L'objectif principal de cet ordonnanceur est de maximiser l'utilisation des ressources et du débit dans un environnement de cluster. Cet algorithme d'ordonnancement [Dong et al., 2016] peut assurer une gestion équitable des ressources de calcul parmi un grand nombre d'utilisateurs. Il utilise des files d'attente au lieu de pools contrairement à l'ordonnanceur équitable et chaque file d'attente sera affectée à une organisation

après que les ressources se soient réparties entre ces files d'attente.

Afin de contrôler les files d'attente, un mécanisme de sécurité est créé pour garantir que chaque organisation puisse accéder à seulement une seule de ses files d'attente. Cet ordonnanceur offre une garantie de capacité minimale en limitant l'exécution des tâches et des travaux à partir d'une seule file d'attente. Lorsque de nouveaux travaux arrivent en file d'attente, les ressources sont réaffectées à la file d'attente précédente après l'achèvement des tâches en cours d'exécution. L'ordonnanceur Capacité permet l'ordonnancement des travaux en fonction de la priorité dans la file d'attente de l'organisation.

Avantages

— La stratégie d'ordonnancement Capacité maximise l'utilisation des ressources et le débit dans l'environnement du cluster.
— Il garantit la réutilisation des travaux inutilisés dans les files d'attente.
— Il prend également en charge les fonctionnalités de files d'attente hiérarchiques, l'élasticité et l'opérabilité.
— Il peut allouer et contrôler la mémoire en fonction des ressources matérielles disponibles.

Inconvénients

— L'ordonnanceur Capacité est le plus complexe parmi les trois ordonnanceurs expliqués ci-dessus.
— Il est difficile de bien choisir les files d'attente adéquates.
— Il a certaines limitations sur les travaux en attente pour assurer la stabilité et l'équité du cluster à partir d'une file d'attente et d'un utilisateur unique.

4. **l'ordonnanceur Delay :** L'ordonnancement équitable [Xie et al., 2017] [Dai et Bensaou, 2016] a été conçue pour un partage équitable de la capacité entre tous les utilisateurs, mais il y a deux problèmes : les slots collantes et l'ordonnancement de l'tête de liste se pose à un moment de partage équitable. Pour surmonter ces retardateurs à deux problèmes, le planificateur est développé. Cet ordonnancement [Xie et al., 2017] est introduit en appliquant des modifications à MapReduce avec la localité de données pour obtenir de meilleures performances et le temps de réponse le plus bas pour la tâche Map.

Facebook utilise la même approche pour obtenir la localité dans le cluster Ha-

doop par l'approche d'attente. Selon cette stratégie, si les données ne sont pas disponibles pour la tâche, le task tracker attendra une durée fixe. L'ordonnanceur vérifie la taille du travail s'il y a une demande d'allocation de tâches à partir du nœud local. L'ordonnanceur ignorera le travail trop court et recherchera le travail suivant disponible à exécuter. Si le processus de saut de travail continue pendant longtemps, il crée une tâche non locale pour éviter la famine. L'ordonnanceur Delay trie les problèmes de localisation qui surviennent à un moment de partage équitable, comme les slots collantes et l'ordonnancement de l'entête de liste.

Avantages

— Simple.
— Il n'a pas de surcharge pour les calculs complexes.
— Il résout également le problème de localité.

Inconvénients

— La technique d'ordonnancement Delay n'est pas efficace lorsque la majeure partie des tâches est beaucoup plus qu'un travail moyen.
— Il y a des slots limités par noeud.

5. **L'ordonnanceur Matchmaking :** L'ordonnancement Matchmaking [Gautam et al., 2015] se concentre sur l'amélioration de la localisation des données des tâches map. Il donne une chance égale à chaque nœud esclave de saisir des tâches locales avant d'assigner à un nœud esclave des tâches non locales. Une tâche est appelée tâche locale qui s'exécute sur un noeud où ses données sont disponibles.

   Cet ordonnanceur essaie de rechercher des correspondances comme un nœud esclave contient des données d'entrée, pour chaque tâche map qui ne sont pas affectées. Chaque nœud est marqué par un marqueur de localité pour confirmer que chaque nœud procure une chance égale de saisir des tâches locales. Cet ordonnancement correspond au temps de réponse le plus bas mais à la plus grande localité de données pour les tâches map. Son ordonnancement des travaux strict et modéré pour l'exécution des travaux contrairement à la politique d'ordonnancement FIFO.

   Avantages

— Dans la technique de planification de matchmaking, la localité de données la plus élevée est atteinte.
— Il fournit également une utilisation élevée du cluster.

6. **L'ordonnanceur LATE (Longest Approximate Time to End)** L'objectif principal de LATE [Gautam et al., 2015] est d'optimiser les performances des travaux et de minimiser le temps de réponse des travaux. Lorsque les petits travaux sont en cours d'execution, le temps de réponse est important. Cependant, les longs travaux sont très lents à cause de nombreux problèmes tels que le grand nombre de processus lent en arrière-plan, la charge CPU, l'indisponibilité des ressources, etc.

   Le travail de l'ordonnanceur LATE est de trouver le processus qui s'exécute avec une vitesse lente dans un cluster et de créer un processus équivalent en arrière-plan en tant que sauvegarde, de tels types de processus sont appelés exécution spéculative de tâches. Cet ordonnanceur est très robuste pour l'hétérogénéité mais il ne fournit pas de garantie de fiabilité. Par défaut, l'ordonnanceur prend en charge les clusters homogènes.

   Avantages
   — L'ordonnanceur LATE optimise les performances des travaux et minimise le temps de réponse du travail autant que possible.
   — Il est très robuste dans le sens de l'hétérogénéité.

   Inconvénients
   — L'ordonnanceur LATE ne garantit pas la fiabilité.
   — Cette technique d'ordonnancement ne fournit pas les garanties de fiabilité.

7. **L'ordonnanceur conscient des ressources** L'ordonnanceur [Cheng et al., 2015] [Yang et al., 2016] est utilisé pour minimiser l'utilisation des ressources. Alors que dans d'autres ordonnanceurs comme l'ordonnanceur équitable, FIFO, Capacité, il est nécessaire que l'administrateur affecte d'abord des travaux à une file d'attente, puis s'assure que les ressources sont partagées manuellement. Dans cet ordonnancement [Shyam Deshmukh, 2013], diverses ressources telles que l'utilisation du réseau, l'utilisation de la mémoire, l'utilisation du processeur, l'utilisation des E / S et l'utilisation des disques ont été partagées plus efficacement.

Aujourd'hui, cette technique apparaît comme un défi de recherche dans le cloud computing. Dans ce plan, l'ordonnancement est accomplie par deux nœuds nommés nœud maître et nœud de travail, également appelés Job Tracker et Task Tracker, respectivement. Job Tracker gère la liste des tâches allouées à chaque Task Tracker, les états des Task Trackers dans le cluster et la file d'attente des tâches en cours tandis que Task Tracker est responsable de l'exécution de chaque tâche configurée avec le nombre maximum de slots disponibles. Deux métriques de ressources "filtrage de slot libre" et "publication de slot libre dynamique" sont utilisées dans cet ordonnanceur. L'ordonnanceur calcule les slots dynamiquement à l'aide de la publication de slot libre dynamique acquise à partir de chaque nœud au lieu d'un nombre fixe d'emplacements de calcul configurés avec chaque nœud Task Tracker. La publicité des emplacements libres dynamiques utilisait ces emplacements libres dynamiquement selon la demande. Le nombre maximal de slots est fixé par nœud en fonction de la disponibilité des ressources.

Avantages

— L'ordonnanceur conscient des ressources aide dans les performances de la gestion des travaux.

— Cet ordonnanceur a également une meilleure utilisation des ressources dans le cluster. [Cheng et al., 2015]

Inconvénients

— L'ordonnanceur fournir un soutien insuffisant dans la préemption des tâches reduce.

— Dans cette technique d'ordonnancement, il existe un besoin de capacités supplémentaires pour gérer les goulots d'étranglement du réseau.

8. **L'ordonnanceur de contraintes de délai** Dans cette stratégie [Cheng et al., 2015] [Dai et Bensaou, 2016], l'utilisateur spécifie les contraintes de délai au moment de l'ordonnancement des travaux, s'assure que les travaux ordonnancés pour l'exécution respectent les délais. Il se concentre sur les problèmes de délais et augmente l'utilisation du système. Son traitement de l'exigence de délai par un modèle de coût de l'exécution du travail qui tient compte des paramètres tels que la taille des données d'entrée, la distribution des données, le temps d'exécution de map et reduce, etc.

Avantages

— L'ordonnanceur de délai met davantage l'accent sur l'optimisation de l'implémentation de Hadoop.

— Cette technique d'ordonnancement augmente également l'utilisation du système.

Inconvénients

— Il y a une restriction sur les noeuds ce qui entraîne des coûts.

— Quelques restrictions ou problèmes de délai qui sont spécifiés par l'utilisateur avec des travaux.

[Wang et al., 2013] Utilisent les informations sur la tâche currente et l'état du système pour effectuer des simulations en ligne en même temps dans Hadoop et prédire avec précision les événements futurs, par exemple quand un travail est terminé ou quand les nœuds de données locaux seront disponibles. Ces prédictions peuvent ensuite être utilisées pour prendre des décisions d'ordonnancement plus efficaces.

Le framework se compose de deux éléments : (i) un prédicteur de tâche qui prédit les temps d'exécution des travaux en fonction de données historiques du même type que les tâches ; et (ii) un simulateur de travaux qui instancie l'ordonnanceur réel dans un environnement simulé et prédit les décisions d'ordonnancements attendues pour toutes les tâches. Cette méthode a deux limitations. Premièrement, il ne peut prédire l'heure d'exécution que pour les travaux soumis au démarrage de la simulation. Deuxièmement, la prédiction du temps d'exécution peut ne pas être très précise car elle utilise un modèle de régression linéaire simple basé uniquement sur la taille des données d'entrée. Dans notre méthode [Merabet et al., 2018], nous utilisons un modèle plus complexe pour la prédiction.

[Liu et al., 2015] introduisent une nouvelle technique basée sur l'ordonnancement de priorité dynamique et un modèle de prédiction en temps réel. L'ordonnancement de priorité dynamique utilise un algorithme de localisation de données avec un coût minimal avec un poids pour des travaux de taille différente et un modèle de prédiction en temps réel qui peut prédire l'utilisation de ressources de tâches non exécutées en calculant des tâches en cours. L'utilisation des ressources contient le processeur, la mémoire et le réseau. Cette technique corrompt l'équilibrage de charge entre les

nœuds et induit une dégradation des performances.

[Lee et al., 2016] proposent hybrid job-driven scheduling scheme (JoSS). Il fournit un ordonnancement au niveau travaux, niveau tâches map et niveau tâches reduce. L'objectif est d'améliorer la localisation des données pour les tâches map et les tâches reduce, d'éviter la famine des travaux et d'améliorer les performances d'exécution des tâches.

3.4.2 Ordonnancement des taches

Niveau de tâche Map

L'ordonnanceur de taches fourni par défaut avec Hadoop prend en compte la localité de données en introduisant une hiérarchie de localités, à savoir la localité de nœud, la localité de rack et les nœuds non locaux, il sélectionne de façon aléatoire une tâche locale (parmi toutes les tâches locales disponibles) ou une tâche non locale lorsqu'un nœud est disponible. Des travaux de recherche ont trouvé [Ibrahim et al., 2012] [Zhang et al., 2011] qu'un tel arrangement conduit à un grand nombre d'exécutions de tâches non locales par l'ordonnanceur.

Une variation de l'algorithme Delay [Zaharia et al., 2010] appelé Random Peeking Scheduling [Tan et al., 2012] a été proposée ; pour ordonnancer les tâches map afin d'améliorer la localité de données, elle retarde l'exécution d'une tâche map en fonction de certaines conditions du contexte pour améliorer la localisation des données. En particulier, pour retarder un travail, l'algorithme considère (i) le pourcentage de nœuds qui ont des données locales pour ce travail, (ii) le nombre de tâches map restantes de ce travail, et (iii) le nombre total de slots map disponibles. Si de nombreux nœuds ont des données locales et qu'un grand nombre de slots sont disponibles, l'exécution des tâches map est retardée. Cependant, si les tâches map non traitées d'un travail sont plus nombreuses que les slots map disponibles, alors sa tâche map n'est pas retardée et elle est affecter a un nœud non local. Se référer à [Ibrahim et al., 2012], qui met en évidence diverses conséquences de la sélection de tâches locales aléatoires non sensibles à l'environnement sur la localité de données globale d'un système.

Pour réduire les exécutions de tâches non locales, [Zhang et al., 2011] proposent un algorithme Next-K Scheduling (NKS) prospectif. Cet algorithme ignore l'ordonnance-

ment des tâches qui n'ont pas de données locales sur le nœud actuellement disponible mais qui ont des données locales sur l'un des nœuds k à venir. Les probabilités d'exécution de chaque tâche en attente sur un nœud disponible sont calculées en fonction du niveau de localité de données sur le nœud actuellement disponible, du niveau de localité de données sur les k prochains nœuds prédits et de l'ordre de traversée des k nœuds suivants. Les trois niveaux de localité de données (nœud, rack et hors rack) sont pris en compte lors du calcul de cette probabilité. Ensuite, la tâche avec la probabilité la plus élevée est affectée sur un nœud disponible alors que d'autres tâches avec des probabilités faibles sont réservées pour une exécution locale de données ultérieure.

[Bu et al., 2013] présentent une stratégie d'ordonnancement des tâches basée sur la localisation des données pour un cluster MapReduce virtuel. Il comprend une politique d'ordonnancement prenant en compte les interférences, basée sur un modèle de prédiction de performance de tâche, et un algorithme d'ordonnancement de délai adaptatif pour l'amélioration de la localisation des données. Les résultats expérimentaux montrent qu'il peut améliorer considérablement le débit du système par rapport aux autres ordonnanceurs.

[Weina et Lei, 2016] proposent Joint Scheduler, un algorithme d'ordonnancement / routage qui traite le problème de la localité de données pour l'ordonnancement des tâches en acheminant les données à l'avance via le réseau de communication dans un cluster MapReduce. Les résultats de la simulation montrent que Joint Scheduler améliore le débit et le délai par rapport à l'ordonnanceur Fair avec l'ordonnanceur delay.

[Sun et al., 2014], [Sun et al., 2016] étudient le problème du délai d'accès aux données causé par l'affectation de certaines tâches map sur des nœuds sans données d'entrée. Ils construisent HPSO (High Performance Scheduling Optimizer), un ordonnanceur de tâches basé sur un service de préchargement pour améliorer la localisation des données. L'idée de base est de prédire les nœuds les plus appropriés auxquels les futures tâches map doivent être assignées et de précharge les données d'entrée dans la mémoire sans retard au démarrage de nouvelles tâches. Les résultats expérimentaux montrent que la méthode réduit le temps s'execution ses tâches map qui provoquant des retards de données et améliore les performances. Cependant, l'inconvénient de HPSO est la précision de préchargement qui affecte les performances, car ils prédisent le temps d'exécution en calculant le temps restant des tâches map exécutées sur les

nœuds.

[Benifa et Dejey, 2017] présentent une stratégie d'ordonnancement nommée *Efficient Locality and Replica Aware Scheduling (ELRAS)* intégré à autonomous replication scheme (ARS) est proposé pour améliorer la localité de données et fonctionne de manière cohérente dans l'environnement hétérogène. Les résultats expérimentaux montrent qu'il peut améliorer le débit d'un facteur d'environ 2 par rapport au FIFO existant et qu'il fournit également une localisation de données proche de l'optimum, réduit le temps d'exécution et l'utilisation efficace des ressources.

Manifestement, les méthodes d'ordonnancement mentionnées ci-dessus sauf [Sun et al., 2016] présentent l'inconvénient commun, qui ne tiennent pas compte de l'impact du délai de transfert de données sur le coût du temps d'exécution de la phase map.

Dans [Merabet et al., 2018] nous proposons un ordonnanceur prédictive de tâches map qui affecte les tâches map les plus appropriées aux nœuds à l'avance. Pour à cela, un modèle de régression linéaire a été utilisé pour la prédiction et un algorithme basé sur la localisation des données pour l'ordonnancement des tâches. Les résultats expérimentaux montrent que la méthode peut grandement améliorer à la fois la localisation des données et le temps d'exécution des tâches map.

Niveau de tâche Reduce

La tâche reduce implique la collecte de données à partir des nœuds map. Ainsi certains problèmes de congestion du réseau et de performances peuvent survenir à cause d'un ordonnancement aléatoire des tâches reduce par le Job tracker d'Hadoop. Si le nœud fournissant une entrée à une tâche reduce donnée qui n'est pas à proximité, alors un réarrangement de données important est nécessaire pour obtenir des données vers le nœud reduce, ce qui entraîne un trafic réseau élevé. Un autre problème est celui de l'asymétrie de partitionnement due à une distribution déséquilibrée des clés intermédiaires entre les nœuds, ce qui entraîne un volume élevé de données pour certaines tâches reduce. Ces deux facteurs peuvent avoir un impact sur les performances de l'application.

Pour résoudre ces problèmes, [Hammoud et al., 2012] proposent *Center-of-Gravity Reduce Scheduler (CoGRS)*, un ordonnanceur de tâches reduce qui prend en compte la localité de données et l'asymétrie. CoGRS affecte chaque tâche Reduce à son nœud de centre de gravité. Des expériences avec CoGRS montrent que cet algorithme minimise

le trafic réseau de manière significative en augmentant les données locales de nœud. Cependant, il ne prend pas en compte la congestion du réseau et l'utilisation des slots dans les clusters dans lesquels trop de travaux s'exécutent simultanément et la congestion du réseau est inévitable.

Le Wait scheduler est une partie de l'algorithme d'ordonnancement du couplage [Tan et al., 2012], utilise le délai pour éviter l'asymétrie de données pour les tâches reduce et améliorer la localisation des données. Une fois que le couplage des tâches reduce avec les tâches map est effectué pour éviter la privation de ressources et qu'une tâche reduce est sélectionnée, il est attendu jusqu'à ce qu'un nœud central aux données intermédiaires générées par les tâches map soit disponible.

[Arslan et al., 2014] proposent l'algorithme LONARS qui permet l'ordonnancement des tâches reduce pour améliorer à la fois la localisation des données et le trafic réseau des applications de données intensives. L'algorithme aide à choisir le task tracker optimal pour toutes les tâches en attente. Un signal de présence est envoyé par tous les task trackers lorsqu'un travail est prêt à ordonnancer la tâche reduce. Une fonction de coût est définie pour déterminer le coût de donner une tâche reduce à chaque task tracker. Les task trackers sont divisés en groupes pour chaque tâche reduce en fonction de la fonction de coût en utilisant le groupement K-MEANS. Un task tracker avec une fonction de coût inférieur est le meilleur candidat pour la tâche reduce. L'algorithme LONARS tente de réduire le trafic réseau par rapport à [Hammoud et al., 2012].

3.5 Comparaison des approches d'ordonnancement

Dans le but de comparer les approches d'ordonnancement basées sur la localité des données, nous allons proposer les critères d'évaluations suivantes :

— **Technique d'ordonnancement :** Différentes techniques d'ordonnancement peuvent être utilisées pour affecter les taches aux noeuds de calcul telles que le calcule d'une fonction de coût, utilisation d'un modèle de prédiction ou d'autres techniques.

— **Objectifs d'optimisation :** Dans un ordonnancement on cherche généralement a optimiser un ou plusieurs objectifs (ou attributs de qualité) cités dans la section 2.

— **Entités d'ordonnancement :** L'ordonnancement peut se faire au niveau travail(job) ou au niveau taches map ou au niveau reduce.
— **Méthode de prédiction :** la prédiction peut suivre un modèle de coût ou un modèle de d'apprentissage (régression linéaire, réseau de neurones ou régression de vecteur de support (SVR)).
— **Environnement :** Un cluster peut être soit composé de machines de configuration identiques (homogènes) ou différentes (hétérogènes).

Critères Travaux	**La Technique d'ordonnancement**	**Les objectifs d'optimisation**	**Les entités d'ordonnancement**	**La méthode de prédiction**	**Environnement**
[Zaharia et al., 2010]	ordonnancement avec Délai	Localité de données et équité	Job		Cluster homogène
[Wang et al., 2013]	prédicteur statistique basée sur l'historique et un Simulateur en ligne des travaux	Localité de données	Job	régression linéaire sur la taille des données	Cluster homogène
[Liu et al., 2015]	algorithme Localité de données avec coût minimum et un modèle de prédiction	Localité de données et équité	Map	Modèle de coût basé sur CPU, mémoire et réseau	Cluster homogène
[Hammoud et al., 2012]	Calcul de Centre de gravité et le Distance Réseau total pondéré	Localité de données et inclinaison de partitionnement	Reduce		Cluster homogène
[Arslan et al., 2014]	Fonction de coût	Localité de données et le trafic réseau	Reduce		Cluster homogène
[Sun et al., 2016]	technique de pré-chargement de données et une fonction de prédiction	Localité de données	Map	Calcule Le temps restant basé sur la lecture des données d'entrée.	Cluster homogène

Critères Travaux	**La Technique d'ordonnancement**	**Les objectifs d'optimisation**	**Les entités d'ordonnancement**	**La méthode de prédiction**	**Environnement**
[Lee et al., 2016]	classification des travaux selon leur type et échelle.	Localité de données	Job Map Reduce		cluster en louant plusieurs serveurs privés virtuels
[Benifa et Dejey, 2017]	mécanisme de réplication	Localité et réplication de données	Job		Clusters hétérogènes
[Merabet et al., 2018]	algorithme ordonnancement basé sur la localité de données, un modèle de prédiction et module de pré-chargement de données	Localité de données	Map	réseau de neurones sur la taille des données et le nombre de taches.	Clusters homogène

TABLE 3.1 – *Comparaison des approches d'ordonnancement.*

3.6 CONCLUSION

La localisation des données est vivement souhaitable pour les travaux qui traitent un volume de données important, car cela minimise le transfert de données sur le réseau, ce qui améliore le temps de réponse.

Dans ce chapitre, nous avons présenté une classification des ordonnanceurs des tâches dans l'environnement Hadoop qui exploitent la localisation des données pour améliorer le temps d'exécution total des travaux. Ensuite, nous avons comparé les principales approches qui ont été proposées dans cet axe de recherche.

4 Un ordonnanceur prédictive de tâches map basé sur l'optimisation de la localité de données.

4.1 Introduction

Hadoop émerge comme un Framework important pour traiter avec des programmes parallèles à grande échelle de données. L'amélioration de la localisation des données est un moyen efficace d'accélérer ces programmes parallèles. Cependant, nous ne pouvons pas charger toutes les données en mémoire en raison de la limitation de la capacité de celle-ci.

Le pré-chargement est un mécanisme intéressant pour une telle situation, qui peut masquer la latence d'accès aux données en chevauchant simultanément le transfert de données et leur traitement. Cependant, le problème est de déterminer comment nous pouvons savoir exactement quelles données doivent être pré-chargées en mémoire.

Dans ce chapitre, nous proposons une approche d'ordonnancement prédictif qui pré-charge les données d'entrée des tâches map sur les nœuds avant le démarrage du traitement. Nous utilisons un modèle de réseau de neurones pour la prédiction et un algorithme basé sur la localité des données pour l'ordonnancement des tâches.

4.2 Concepts clés du système

Ordonnancement de taches

Il y un problème d'ordonnancement quand un ensemble de travaux est à réaliser, que cette réalisation est décomposable en tâches, que le problème consiste à définir

la localisation temporelle des tâches et/ou la manière de leur affecter les moyens nécessaires. Autrement dit, il s'agit d'allouer des ressources aux activités et d'affecter à chaque opération une date de démarrage. La problématique de l'ordonnancement de tâches fait partie des problèmes appelés « optimisation sous contraintes » (en anglais : CSP, Constraint Satisfaction Problem).

On distingue trois catégories de problèmes d'ordonnancement [ilo, 1998] :

Les problèmes d'ordonnancement pur : Dans ce type de problème, on sait quelle ressource est utilisée par quelle activité et en quelle quantité. Les capacités des ressources sont aussi connues, bien qu'elles puissent varier avec le temps. Le problème est de placer les activités dans le temps sans excéder les capacités des ressources disponibles. L'exemple classique du problème d'ordonnancement pur est le « Job shop scheduling problem »

*Les problèmes d'allocation de ressources pures :*Dans ce cas, l'instant d'exécution de chaque tâche est connu ; il s'agit seulement d'allouer les bonnes ressources, de façon à assurer qu'à tout instant, leur disponibilité dépasse la demande. A titre d'exemple, on peut citer l'allocation de personnel et la gestion des lits au niveau du bloc opératoire d'un hôpital.

Les problèmes conjoints d'ordonnancement et d'allocation de ressources : Il existe un degré de liberté aussi bien en ce qui concerne les activités à effectuer que les ressources à utiliser.

Localité des données

Dans le système de fichiers distribué comme le HDFS d'Hadoop, plusieurs copies de chaque bloc de données sont stockées sur des machines différentes (généralement trois dans Hadoop). En raison de la réplication des données, les chances de trouver un nœud ayant une copie locale des données deviennent élevées, de sorte que l'ordonnanceur tente d'affecter une tâche map aussi proche que possible des données. Hadoop considère plusieurs niveaux de localités de données lors de l'ordonnancement des tâches. Il suppose que les nœuds sont répartis sur plusieurs racks à travers le cluster. Il calcule la bande passante entre deux nœuds en fonction de leur distance dans la topologie, ce qui implique que deux nœuds sur le même rack ont une bande passante supérieure à ceux qui sont hors-rack. L'ordonnanceur utilise ces informations lors de l'ordonnancement des tâches pour améliorer la localisation des données.

Tout d'abord, il tente d'atteindre la localité du nœud en exécutant une tâche map sur un nœud contenant une copie des données correspondantes. Un tel nœud est appelé le nœud local. Si ce n'est pas possible, il tente de localiser le rack en recherchant une autre machine dans le même rack. Ce nœud est dit rack local. Au pire, il choisit la non-localisation en affectant la tâche map sur une machine inactive dans la même zone de commutation réseau, appelée noeud non local.

Préchargement

La pré-chargement (*prefetching*) est une technique utilisée pour résoudre les deux problèmes suivants :

Le premier problème est que, lorsqu'un travail s'exécute en premier, aucun bloc d'entrée n'est mis en cache en mémoire pour le travail. Le deuxième problème est que certains blocs libérés lors du tour précédent à cause d'une insuffisance de mémoire seront utilisés lors du tour suivant. Nous utilisons la technologie de pré-chargement pour réduire le coût de chargement des données libérés. En règle générale, dans un travail MapReduce, le bloc de données ne sera traité qu'une seule fois dans un cycle. Ainsi, les blocs traités peuvent être remplacés si la mémoire est insuffisante pour mettre en cache de nouveaux blocs.

Comme nous avons décrit dans le chapitre III, les données d'entrée d'une tâche doivent être extraites en mémoire avant l'ordonnancement de la tâche sur le noeud. Cela posera un problème que les données pré-chargées et la tâche correspondante ne soient pas situées dans le même nœud. Ceci peut entraîner une opération de pré-chargement qui ne peut pas accélérer l'exécution du programme, mais uniquement augmenter la charge du système.

La plupart des travaux récents utilisent des méthodes de pré-chargement reposant sur la prédiction [Chen et al., 2012]. Avant l'opération de pré-chargement, le nœud doit prédire quelle tâche map sera assignée au noeud et décider quel bloc correspond aux données d'entrée de cette tâche map. Selon cette prédiction, ce nœud pré-charge les données auxquelles la prochaine tâche map accédera. Ces stratégies de pré-chargement sont basées sur la précision de la prédiction. Une prédiction précise et efficace permet de pré-chargé les bonnes données qui seront utilisées dans le travail. Un mauvais résultat de la prédiction entraînera l'inutilisation des données pré-chargées

par les travaux à venir. Cela entraîne non seulement un échec de pré-chargement, mais gaspille également la bande passante et les ressources du disque dur.

4.3 ARCHITECTURE DU SYSTÈME

Dans cette section, nous présentons l' architecture du système de notre approche qui améliore la localisation des données dans les clusters MapReduce. Le but est de concevoir un ordonnanceur efficace de tâches map en utilisant le pré-chargement des données et la prédiction du temps d'exécution [Merabet et al., 2018]. Lorsqu'un nouveau travail est soumis au Job Tracker :

1. L'analyseur de tâches collecte des informations sur la taille des données d'entrée et le nombre de tâches simultanées sur chaque nœud.
2. Chaque Task Tracker a deux agents : un prédicteur de temps d'exécution et un agent de préchargement qui soumet périodiquement les temps d'exécution estimés des tâches en cours d'exécution sur le nœud et les données d'entrée locales. Sur la base de ces informations, l'ordonnanceur de tâches prend des décisions d'ordonnancement.
3. Le jobTracker déclenche le gestionnaire de pré-chargement qui force les agents de pré-chargement à charger les données d'entrée dans les noeuds sur lesquels la tâche map future sera traitée.
4. Lorsque le jobTracker reçoit une pulsation d'un nœud taskTracker. Il attribue la première tâche dans la file d'attente à ce noeud.

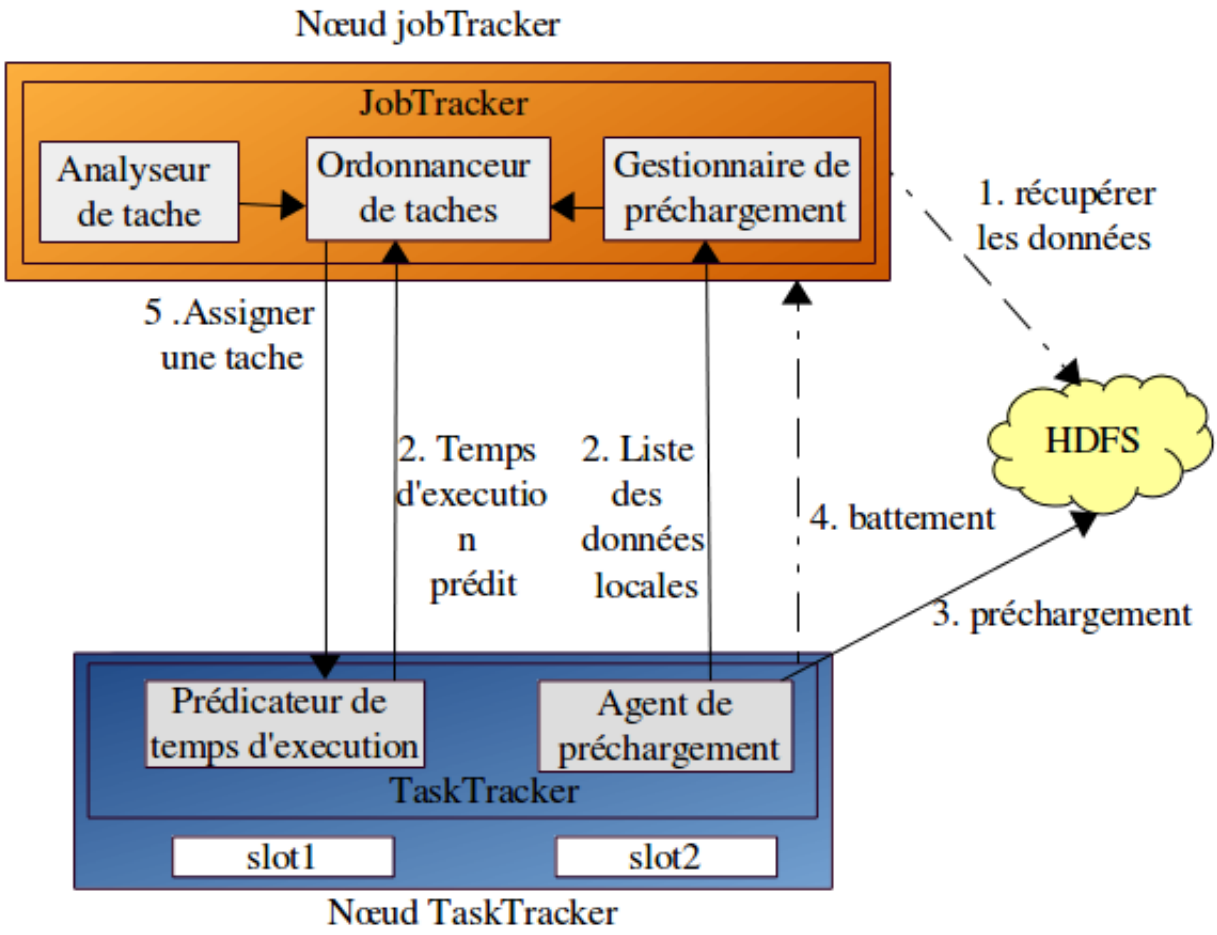

FIGURE 4.1 – *L'architecture de système*

La figure 4.1 montre l'architecture du système qui comprend deux modules principaux : L'ordonnanceur de tâches et le gestionnaire de pré-chargement qui sont détaillés ci-dessous.

4.3.1 L'ordonnanceur des tâches

Son rôle principal consiste à affecter les tâches map aux nœuds TaskTracker les plus proches des données d'entrée. Pour cela, il utilise les informations envoyées par les agents prédictifs et de pré-chargement pour créer un schéma d'ordonnancement pour les tâches map.

Le noeud Job Tracker comprend un analyseur de tâches qui exécute une tâche map et extrait les paramètres pour chaque travail soumis. Ensuite, les agents prédictifs de chaque nœud utilisent ces paramètres pour prédire le temps d'exécution d'une

tâche map. L'ordonnanceur de tâches prédit la séquence des emplacements qui seront bientôt libres sur tous les nœuds du cluster.

4.3.2 Le gestionnaire de pré-chargement

Le module de pré-chargement contrôle les agents de pré-chargement dans chaque nœud TaskTracker et déclenche des actions de préc-hargement des données en entrée. Le gestionnaire de pré-chargement collecte des informations sur toutes les données d'entrée locales fournies par chaque agent et construit une liste de données d'entrée des blocs de données présentes dans tous les nœuds TaskTracker.

D'abord, cette liste aide l'ordonnanceur à prendre des décisions d'ordonnancement afin d'augmenter le nombre de données traitées localement.

Ensuite, le gestionnaire déclenche les agents de pré-chargement pour pré-charger les données requises par les tâches avant l'ordonnancement.

4.3.3 L'analyseur de tâches

L'analyseur de tâches exécute une étape d'apprentissage dans laquelle il lance l'exécution d'un ensemble de tâches map à partir du travail en cours avec différents paramètres tels que la taille des données d'entrée et le nombre de tâches map simultanées. Ensuite, il collecte les valeurs de temps d'exécution des tâches map. Les résultats de cette étape seront utiles pour prédire le temps d'exécution des nouvelles tâches map.

4.3.4 Temps d'exécution de la tâche map

Dans la littérature, il existe deux modèles de prédiction :

Modèle statistique (performance) :

Utilise les traces historiques collectées sur un cluster donné pour prédire le temps d'exécution à l'aide de modéles statistiques ([Herodotos et al., 2011] ; [Herodotos]. [Lin et al., 2012] ; [Ge et al., 2013]).

Modèle de coût :

Divise le traitement en plusieurs étapes. Le coût de chaque étape est calculé par une formule mathématique ([Liu et al., 2015] ; [Sun et al., 2016]).

Notre objectif principal est de prédire le temps d'exécution d'une tâche map à partir des résultats d'exécution des tâches map précédentes.

Deux caractéristiques principales rendent notre approche réalisable :

— La similarité entre les tâches map d'un même travail : Dans le modèle MapReduce, un élément de données en entrée est traité par la même fonction map dans tous les nœuds TaskTracker.

— La similarité entre des blocs de données d'entrée provenant du même jeu de données : le JobTracker divise les données d'entrée du travail en cours en blocs de données, de sorte que les blocs de données ont le même format.

La plupart des approches utilisent uniquement la taille des données d'entrée pour estimer le temps d'exécution, ce qui affecte la précision des prédictions. L'approche proposée par [Ge et al., 2013] suppose qu'une tâche map est composée des 10 facteurs (voir la figure 4.2).

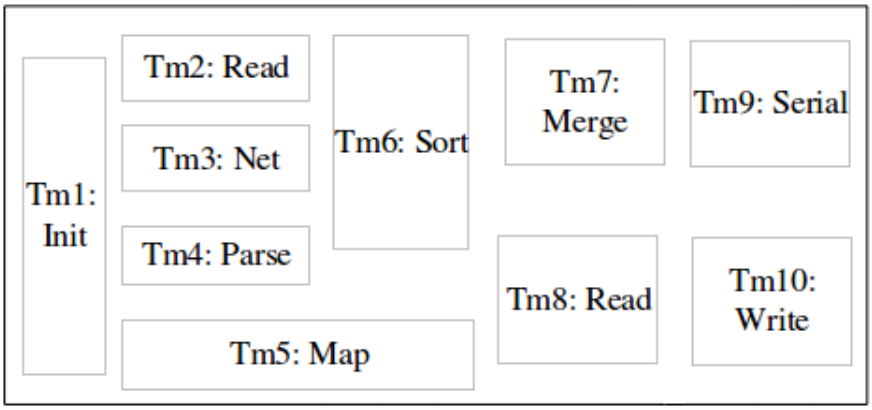

FIGURE 4.2 – *La phase Map [Lin et al., 2012]*

La durée d'exécution totale d'une tâche map peut être représentée par la fonction suivante :

$$T_{map} = F(T_{m1}, T_{m2}, ..., T_{m10}) \tag{4.1}$$

Où T_{map} représente le temps d'exécution totale d'une tâche map.

-

Ces facteurs d'exécution sont affectées par deux propriétés : la taille de l'entrée de données (DataInput) et le nombre de tâches map simultanées sur chaque noeud (NbTasks).

Properties	**Facteurs**
DataInput	$T_{m2}, T_{m3}, T_{m7}, T_{m8}, T_{m10}$
NbTasks	$T_{m2}, T_{m3}, T_{m4}, T_{m5}, T_{m6}, T_{m7}, T_{m8}, T_{m9}$

TABLE 4.1 – *Relation entre les étapes et les propriétés*

.

Dans notre approche, le prédicteur construit un modèle de performance pour les tâches déjà exécutées, puis applique ce modèle pour prédire le temps d'exécution des nouvelles tâches. Le prédicteur observe les valeurs des deux propriétés et le temps d'exécution correspondant pour toutes les tâches précédentes et entraine un modèle de réseau de neurones pour déterminer la corrélation entre elles. Le tableau 4.2 représente le coefficient de corrélation entre Tmap et les propriétés pour plusieurs types de travaux.

Job Name	*WordCount*	*Dedup*	*Project*	*Grep*
R^2	0.9988	0.9812	0.9973	0.9869

TABLE 4.2 – *La corrélation entre Tmap et les propriétés des différents travaux*

Pour choisir le modèle le plus précis possible pour notre prédicateur de taches, nous avons testé les quatre modèles suivants :

1. **Modèle de régression linéaire simple** Ce modèle utilise une seule caractéristique (taille des données d'entrée). Ainsi, il représente une relation linéaire entre les valeurs de cette variable d'entrée et le temps d'exécution des tâches map pour chaque travail :

$$T_{map} = a_0 + a_1 * DataInput \tag{4.2}$$

Où la variable DataInput représente les valeurs de la taille des données en entrée et a_0, a_1 représentent les coefficients de la relation linéaire.

2. **Modèle de régression linéaire multiple** Similaire au modèle précédent mais ajoute une deuxième caractéristique (nombre de tâches map exécutées simultanément par un nœud), le modèle est représenté comme suit :

$$T_{map} = a_0 + a_1 * DataInput + a_2 * NbTasks \tag{4.3}$$

Où la variable NbTasks représente le nombre de tâches map exécuté en parallèle dans un nœud et a_0, a_1, a_2 représentent les coefficients de la relation linéaire.

3. **Modèle de régression par vecteur de support** La régression par vecteur de support (SVR) utilise les mêmes principes que le SVM, à la différence que la sortie est un nombre réel, le modèle est représenté comme suit :

$$T_{map} = \sum_{n=1}^{N} (\alpha_{1n} - \alpha_{1n}^*) * (DataInput_n^{'} DataInput) + \sum_{n=1}^{N} (\alpha_{2n} - \alpha_{2n}^*) * (NbTasks_n^{'} NbTasks) + b \tag{4.4}$$

Où $\alpha_{1n}, \alpha_{1n}^*, \alpha_{2n}, \alpha_{2n}^*$ représentent les multiplicateurs non négatifs pour chaque observation x_n de chaque caractéristique.

4. **Modèle de réseau de neurones** Nous utilisons une classe de modèles de réseaux de neurones appelés Multi-Layer Perceptron (MLP), qui est définis comme suit : la couche d'entrée consiste en un ensemble de neurones $(DataInput, NbTasks)$ représentant les caractéristiques en entrée. Chaque neurone de la couche cachée transforme les valeurs de la couche précédente par une somme linéaire pondérée.

$$w_1 * DataInput + w_2 * NbTasks \tag{4.5}$$

Où w_1, w_2 représentent les poids des caractéristiques, suivie d'une fonction d'activation non linéaire comme la fonction de tan hyperbolique. La couche de sortie reçoit les valeurs de la dernière couche cachée et les transforme en valeurs de sortie.

4.3.5 Ordonnanceur des tâches

L'ordonnanceur de tâches crée une liste des emplacements qui deviendront inoccupés, grâce aux prédictions de temps d'exécution effectuées par le prédicteur dans chaque nœud TaskTracker. Ainsi, il assigne une tâche dans la file d'attente des travaux au premier emplacement de la liste de séquences prédite de l'emplacement. La figure 4.3 montre un exemple de cluster à 3 nœuds avec deux (2) emplacements par nœud pour les tâches map et une file d'attente de huit (8) tâches map :

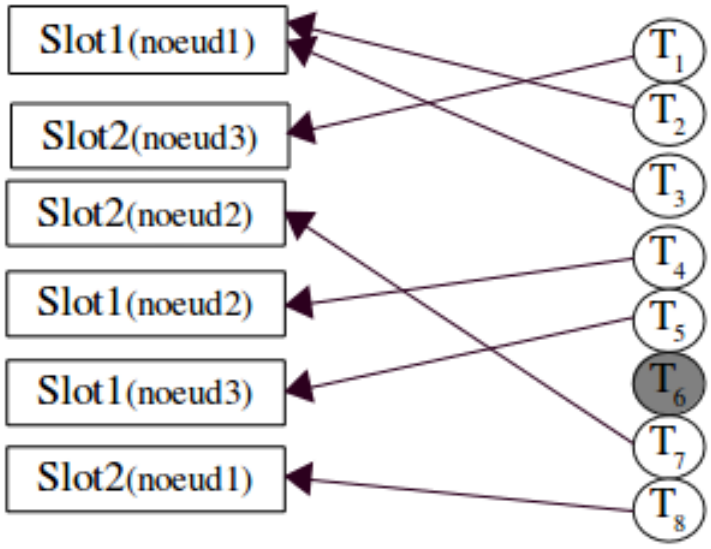

FIGURE 4.3 – *exemple d'ordonnancement.*

L'ordonnanceur déclenche ensuite le gestionnaire de pré-chargement pour charger les données d'entrée des futures tâches map dans les nœuds correspondants. Ainsi, lors du traitement des données d'entrée par une tâche map, les données requises sont déjà stockées dans le nœud qui réduit le temps d'attente pour le transfert de données et limite le trafic réseau.

L'ordonnanceur calcule l'emplacement d'un noeud défini comme le temps maximum nécessaire pour consolider les données d'entrée d'une tâche à partir d'emplacements de stockage. La localité du nœud est définie comme suit :

$$LOC_{t_j} = \max_{h=1,\ldots,H} (d_{h_i} * r_{h_i}) \tag{4.6}$$

où d_{h_i}est la donnée stockée dans le noeud h_i et r_{h_i} est le temps de transfert $h = 1, H$ entre le noeud du t_j et le noeud h_i.

L'algorithme d'ordonnancement est décrit dans Algorithme 1

Algorithme 1 : Algorithme d'ordonnancement

```
input : tab N : la séquence des slots prédits.
        tab J : la liste d'attente des travaux.
begin
  while !(tous les slots ont : nombre de taches en attente< capacité (slot)) ou
    (aucun travail dans J) do
      n : le premier slot de N;
      for j dans J do
        tab : nl les taches de j dont les données sont stockées dans le même nœud
          du slot n;
        tab : rl les taches de j dont les données sont stockées dans le même rack
          que le nœud du slot n ;
        for t dans T_j do
          if t est une tâche nœud-locale then ajouter t a nl;;
          else if t est une tâche rack-locale then
            ajouter t a rl;
        m = (capacité de slot− nombre de tâches en attente)
        if nl <> 0 then
          trier nl par LOCt_i;
          affecter les m premiers taches de nl au slot n;
          déclencher le préchargement des données d'entrée;
          break;
        else if rl <> 0 then
          trier rl par LOCt_i;
          affecter c premiers taches de rl à n ;
          déclencher le préchargement des données d'entrée;
          break;
        else
          affecter une tache aléatoire à n;
          déclencher le préchargement des données d'entrée;
      Rtn =Rtn + Pred(taches affectées à n) ; Rtn : temps restent pour libéré le
        slot n;
      Pred : prédiction de temps d'execution ;
      trier tab N;
```

Pour le premier emplacement n de la séquence d'emplacements prévus, l'algorithme construit deux listes :

— La liste nl des tâches dans la liste des travaux dont les données d'entrée sont stockées dans le même nœud de l'emplacement n.

— La liste rl des tâches dans la liste des travaux dont les données d'entrée sont stockées dans le même rack du nœud de l'emplacement n.

Chaque emplacement d'un nœud donné a la capacité de gérer un certain nombre de tâches map. Ensuite, en fonction de la capacité actuelle de l'emplacement n, nous affectons un groupe de tâches dans l'ordre suivant : Premièrement, les tâches avec la localité la plus élevée dans la liste nl. Si cette liste est vide, nous passons aux tâches avec la localité la plus élevée dans la liste rl. Sinon, des tâches aléatoires sont choisies. Le pré-chargement des données d'entrée sur les nœuds correspondants est déclenché et le temps de traitement restant de l'emplacement n est mis à jour.

Le pré-chargement des données d'entrée sur les nœuds correspondants est déclenché et le temps de traitement restant de l'emplacement n'est pas mis à jour. Ces opérations sont répétées avec le prochain emplacement. L'algorithme se termine si tous les emplacements sont occupés ou s'il n'y a pas de travail dans la liste des travaux.

L'algorithme proposé ne modifie pas la priorité des travaux et tolère les défaillances en raison de la disponibilité des données d'entrée pour les tâches. Notre algorithme d'ordonnancement, comparé à [Wang et al., 2013] et [Sun et al., 2016], prédit avec précision la séquence de libération d'emplacements, qui précharge les données d'entrée sur le même nœud où les tâches futures seront exécutées (voir figure 4.5).

4.3.6 Pré-chargement de données :

La technique de pré-chargement implique le chevauchement de la phase d'exécution de la tâche avec la phase de transfert de données, ce qui améliore les performances d'un cluster MapReduce.

Dans notre approche, le gestionnaire de pré-chargement déclenche le pré-chargement des données d'entrée avant le traitement des taches sur les nœuds du TaskTracker. Il crée une liste de données pour chaque nœud TaskTracker. Cette liste est périodiquement mise à jour par les messages envoyés par les agents de pré-chargement.

La Figure. 4.4 montre la gestion des buffers pour chaque emplacement dans chaque nœud. Chaque emplacement a deux tampons, un buffer pour le bloc de données pour la tâche en cours d'exécution (D2, D5) et un autre buffer pour le bloc suivant de la même tâche ou pour le premier bloc de la tâche suivante (D3, D6). L'avantage de notre méthode de pré-chargement est qu'elle masque la latence de la mémoire en diminuant le temps d'accès aux données par les emplacements tout en économisant l'utilisation de la mémoire puisque chaque emplacement n'utilise que deux buffers.

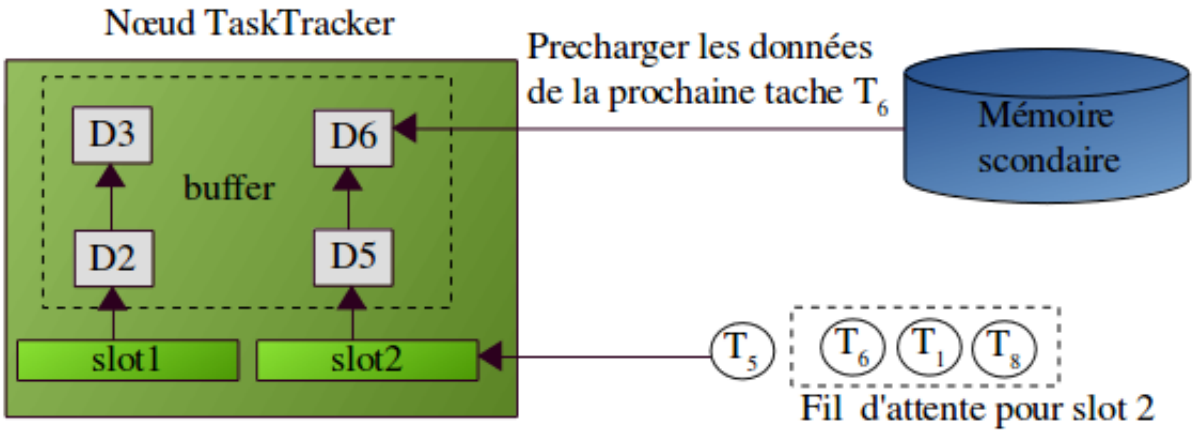

FIGURE 4.4 – *La gestion du buffer*

4.4 ETUDES EXPÉRIMENTALES ET VALIDATION

4.4.1 Environnement expérimental

Nous avons évalué notre ordonnanceur de tâches par rapport a l'ordonnanceur par défaut d'Hadoop (FCFS) et cela en fonction de la précision du prédicteur de tâches et les performances de l'ordonnanceur. Nous avons également étudié l'impact de notre approche sur la localisation des données (data locality). Nous décrivons d'abord notre configuration expérimentale avant de présenter nos résultats. Nous avons utilisé un cluster de cinq (5) nœuds avec un (1) JobTracker comme maître et quatre (4) TaskTrackers comme esclaves. Hedoop 2.7.3 est installé et nous avons configuré chaque TaskTracker avec deux (2) emplacements map et deux (2) emplacements reduce.

Nous effectuons notre expérimentation en utilisant le benchmark wordcount avec trois jobs de différents nombres de tâches map, à savoir 10, 15 et 20. Nous avons également configuré Hadoop pour lancer des tâches reduce uniquement lorsque toutes les tâches map sont terminées. L'exécution spéculative est désactivée. Les autres paramètres ont été définis sur leurs configurations par défaut.

4.4.2 Précision de prédiction du prédicteur de tâche

Nous avons évalué la précision de prédiction du prédicteur de tâche. La figure. 4.5 montre la précision de la prédiction concernant l'exécution des tâches map avec quatre modèles d'apprentissage automatique. Nous pouvons constater à partir de cette expé-

rience que lorsque nous utilisons une régression linéaire multiple [Merabet et al., 2018], la précision de la prédiction est plus élevée. Cependant lorsque nous utilisons une régression linéaire simple, le taux de précision diminue. En effet, deux caractéristiques sont utilisées dans le premier modèle plutôt que d'utiliser qu'une seule caractéristique dans le second. Cependant, le taux de précision de prédiction optimale est obtenu en utilisant le réseau de neurones (ANN) et la régression par vecteur de support (SVR) car ils peuvent apprendre des modèles non linéaires.

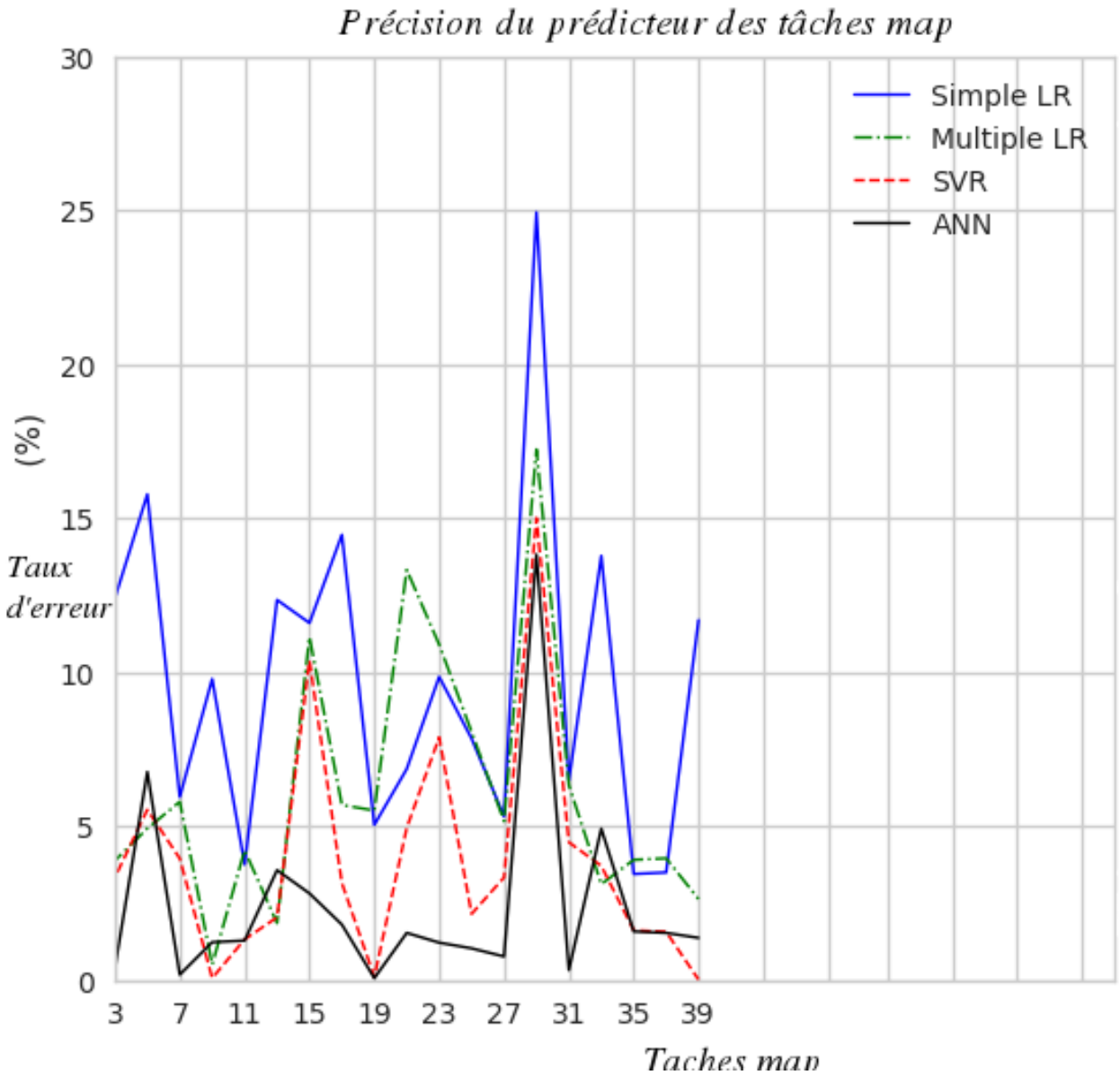

FIGURE 4.5 – *Prédiction de tâches map [Merabet et al., 2018]*

Dans la figure. 4.6 et la figure. 4.7, nous avons comparé les performances de notre ordonnanceur avec l'ordonnanceur par défaut d' Hadoop (basé sur la politique FIFO) en termes de taux de localisation des données et de temps d'exécution.

4.4.3 Localité de données

Comme le montre la figure 4.6, l'ordonnanceur prédictif proposé atteint un taux de localisation des données supérieur à celui de l'ordonnanceur Hadoop pour tous les travaux, avec une moyenne de 78%. Cela est dû au fait que chaque tâche map est affectée au nœud où le jeu de données requis est déjà préchargé grâce à la prédiction du temps d'exécution. Cependant, la localisation des données ne peut dépasser un seuil de 82% en raison de la congestion du réseau induite par une transmission de données élevée.

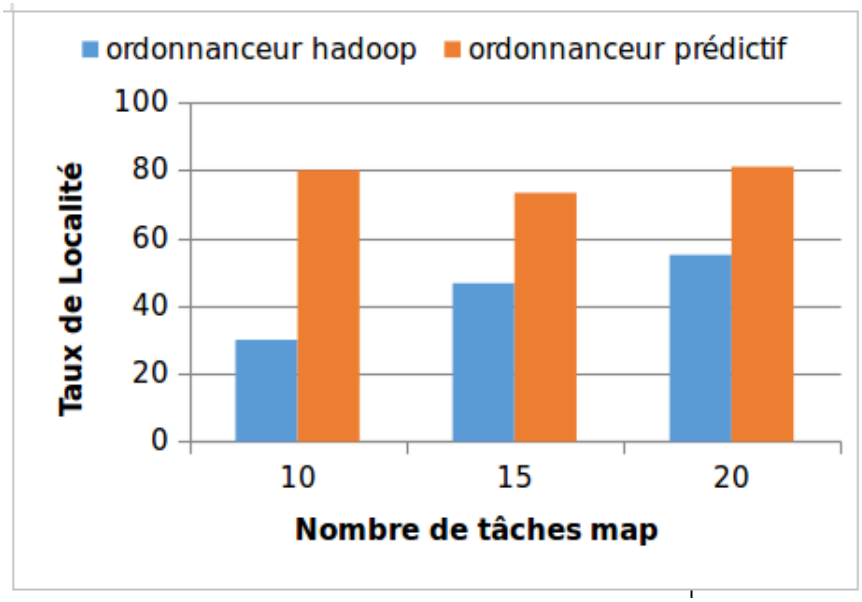

FIGURE 4.6 – *Localité des données [Merabet et al., 2018]*

4.4.4 Temps d'exécution des tâches map

La figure. 4.7 montre que l'ordonnanceur prédictif proposé a sensiblement amélioré le temps d'exécution à mesure que la taille du travail augmente. Cela est dû à l'augmentation de la précision de la prédiction, car nous avons plus de tâches map pour l'étape d'entrainement. Cependant, notre approche est limitée par la capacité de traitement du cluster et le taux d'erreur de prédiction de 5% qui induit des décisions de

préchargement inefficaces. Les résultats de la figure. 4.7 montrent que l'ordonnanceur proposé continuera à bien fonctionner à une plus grande échelle. Bien entendu, le seul moyen d'évaluer de manière concluante les performances à grande échelle consistera à le déployer sur un cluster de grande taille avec un nombre considérable de nœuds et de travaux comportant un nombre important de tâches.

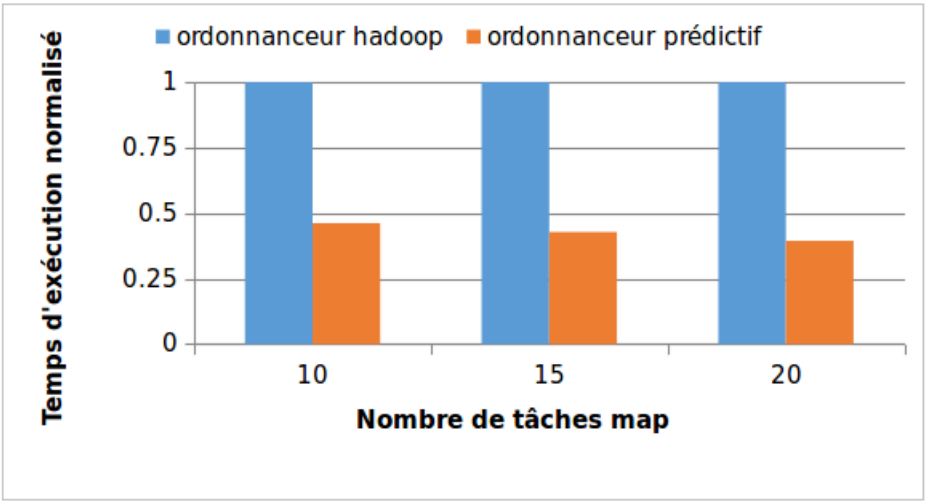

FIGURE 4.7 – *Temps d'exécution total des tâches map [Merabet et al., 2018]*

Notre modèle de prédiction a amélioré le taux d'erreur de prédiction de 10% à 3,6% (voir figure. 4.5). C'est grâce à l'utilisation de modèle de réseau de neurones avec deux caractéristiques. Ainsi, plus nous utilisons de caractéristiques, plus nous faisons un modèle de prédiction mieux adapté au modèle réel avec moins d'erreurs.

4.4.5 Discussion

En termes de performance, l'ordonnanceur proposé fonctionne mieux que celui d'Hadoop et de [Wang et al., 2013]. Le gain de performance provient de l'augmentation du taux de localisation des données en affectant les tâches map aux nœuds où les données d'entrée sont stockées et en utilisant un mécanisme de pré-chargement pour masquer le délai d'accès distant aux données d'entrée.

La principale différence entre notre approche et Hadoop réside dans la politique d'ordonnancement employée. Plus précisément, dans Hadoop, une tâche map est allouée à un TaskTracker lorsque ce dernier indique qu'il dispose d'un emplacement inoccupé par un signal (heartbeat). Cependant, cela augmente le temps d'exécution des tâches et peut entraîner une congestion du réseau en effectuant un accès réseau à distance.

En revanche, notre approche d'ordonnancement prédit les nœuds TaskTracker les plus appropriés pour les tâches avant le déclenchement des pulsations.
L'inconvénient de notre approche c'est qu'elle peut provoquer un déséquilibrage de charge dans le cluster car elle optimise uniquement la localité des données.

4.5 Conclusion

Ce chapitre présente notre ordonnanceur prédictif de tâches map avec préchargement des données qui vise à réduire le temps d'exécution des travaux en augmentant le taux de localisation des données. Les résultats d'expérimentation montrent que notre approche surpasse l'ordonnanceur d'Hadoop et permet d'améliorer la localisation des données d'au moins 73,33%, ce qui améliore considérablement le temps d'exécution total des tâches map de 57% en moyenne.

Conclusion Générale et Perspectives

Conclusion Générale

Aujourd'hui, les applications Bigdata jouent un rôle très important au sein des entreprises génératrices d'un grand volume de données et permettent un réel avantage concurrentiel. Nous assistons au plein développement des frameworks dédiées au Bigdata. Ces frameworks ont pour but d'utiliser des clusters d'une capacité gigantesque de ressources (calcul, stockage, services, etc) qu'on n'aurait jamais pu avoir avec une seule machine et permet de traiter des volumes de données à grande échelle en un temps modéré.

Hadoop est l'un des frameworks les plus utilisés par ces entreprises. Ce succès est dû à son extrême facilité de programmation, de sa rapidité, de son évolutivité et de sa tolérance aux pannes. Cependant, l'adoption de ce framework a engendré l'émergence de nouveaux problèmes, son amélioration s'avère donc primordiale.

Dans le cadre de cette thèse, nous avons abordé l'un des problèmes les plus critiques du fait de domaine du Bigdata, à savoir, l'ordonnancement. En particulier comme les applications Bigdata sont caractérisées d'être à forte intensité de données. En effet, l'ordonnancement des travaux dans un cluster Hadoop est souvent soumis à des contraintes et des exigences liées à l'infrastructure matérielle. Le type de travail, les utilisateurs, la localité des données sont des facteurs importants qui permettent d'optimiser d'une façon significative l'ordonnancement et par conséquent booster les performances des clusters.

Tout d'abord, dans le chapitre 2, nous avons étudié le domaine de Big Data, ses principales caractéristiques, son évolution et les composants qui constituent l'architecture Big Data. Dans le chapitre 3, nous avons décrit l'architecture de base d'Hadoop, son fonctionnement, ses caractéristiques, ainsi que ses problèmes et défis.

Dans le chapitre 4, nous avons défini les attributs de qualité les plus commun pour les algorithmes d'ordonnancement. Ensuite nous avons élaboré un état de l'art détaillé sur

les différentes approches d'ordonnancement qui utilisent la localisation des données. Ensuite nous avons proposé un tableau comparatif des approches selon des critères d'évaluation que nous avons défini et cela afin de montrer clairement notre contribution.

Dans le chapitre 5, nous avons conçu un ordonnanceur prédictif de tâches map en exploitant la localité des données sur un cluster hadoop formé de plusieurs noeuds homogènes. Pour cela nous avons proposé trois composants dans l'architecture d'Hadoop :

1. L'ordonnanceur des tâches qui affecte les tâches map aux nœuds TaskTracker les plus proches des données d'entrée. Il utilise un algorithme qui calcule la localité de chaque nœud en fonction du temps maximum nécessaire pour consolider les données d'entrée d'une tâche à partir d'emplacements de stockage.
2. Le gestionnaire de pré-chargement qui déclenche le pré-chargement des données d'entrée avant le traitement des taches sur les nœuds du TaskTracker. Ce pré-chargement se base sur les prédictions du prédicateur de taches.
3. Le prédicteur de taches qui se base sur un modèle de réseaux de neurones pour prédire le temps d'exécution des nouvelles tâches. Le modèle utilise les valeurs des deux propriétés et le temps d'exécution correspondant pour toutes les tâches précédentes.

Le but final est d'améliorer les performances d'un cluster Hadoop en réduisant le temps d'exécution totale des travaux et le trafic réseau. Les expérimentations que nous avons menées montrent que notre ordonnanceur surpasse celui d'Hadoop. Nous avons amélioré la localisation des données d'au moins 73,33%, ce qui a amélioré considérablement le temps d'exécution totale des tâches map de 57% en moyenne.

Perspectives future

Le travail que nous avons mené dans le cadre de cette thèse nous permet de dégager quelques perspectives futures :

— pour améliorer notre modèle de prédiction nous pouvons considérer d'autres caractéristiques sont utilisés des modèles plus complexes.

— Supporter de nouvelles exigences (équité entre les utilisateurs, equilibrage des charges, etc)

— Considéré des clusters avec des nœuds hétérogènes et une capacité de bande passante limitée.
— Développement d'un algorithme plus complet permettant de prendre en charge l'ordonnancement à la fois des taches map et des taches reduce.

Enfin, nous prévoyons également d'évaluer les performances de notre approche à grande échelle en la déployant sur un grand cluster afin de mettre en valeur notre contribution.

Bibliographie

ILOG Scheduler - Users Manual. S.A., Gentilly, France, 1998.

Engin Arslan, Mrigank Shekhar, et Tevfik Kosar. Locality and network-aware reduce task scheduling for data-intensive applications. *5 th ACM International Workshop on Data-Intensive Computing in the Clouds*, pages 17–24, 2014.

Marcos Assuncao, Rodrigo N. Calheiros, Silvia Bianchi, Marco Netto, et Rajkumar Buyya. Big data computing and clouds : Trends and future directions. 75, 01 2014.

atlas. Atlas. http://atlasexperiment.org/.

Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, et Peter Vajgel. Finding a needle in haystack : Facebook's photo storage. Dans *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 47–60, Berkeley, CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1924943.1924947.

Bibal Benifa et J. V. Dejey. Performance improvement of mapreduce for heterogeneous clusters based on efficient locality and replica aware scheduling (elras) strategy. *Wireless Personal Communications*, 95(3) :2709–2733, 2017.

Mathiya Bhavin J. et Desai Vinodkumar L. Apache hadoop yarn parameter configuration challenges and optimization. Dans *International Conference on Soft-Computing and Network Security (ICSNS -2015)*, pages 25–27, 2015.

Kyoungsoo Bok, Jaemin Hwang, Jongtae Lim, Yeonwoo Kim, et Jaesoo Yoo. An efficient mapreduce scheduling scheme for processing large multimedia data. *Multimedia Tools and Applications*, 76(16) :17273–17296, Aug 2017. ISSN 1573-7721. URL https://doi.org/10.1007/s11042-016-4026-6.

Peter Brucker. *Scheduling Algorithms*. Springer-Verlag, Berlin, Heidelberg, 3rd édition, 2001. ISBN 3540415106.

Xiangping Bu, Jia Rao, et Cheng-zhong Xu. Interference and locality-aware task scheduling for mapreduce applications in virtual clusters. Dans *Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '13, pages 227–238, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1910-2. URL http://doi.acm.org/10.1145/2462902.2462904.

Carlos Castillo. Effective web crawling. 39 :55–56, 06 2005.

Ronnie Chaiken, Bob Jenkins, Pere Larson, Bill Ramsey, Darren Shakib, Simon Weaver, et Jingren Zhou. Scope : Easy and efficient parallel processing of massive data sets. *Proc. VLDB Endow.*, 1(2) :1265–1276, Août 2008. ISSN 2150-8097. URL http://dx.doi.org/10.14778/1454159.1454166.

Soumen Chakrabarti. Data mining for hypertext : A tutorial survey. *SIGKDD Explor. Newsl.*, 1(2) :1–11, Janvier 2000. ISSN 1931-0145. URL http://doi.acm.org/10.1145/846183.846187.

Soumen Chakrabarti, Martin van den Berg, et Byron Dom. Focused crawling : A new approach to topic-specific web resource discovery. *Comput. Netw.*, 31(11-16) :1623–1640, Mai 1999. ISSN 1389-1286. URL https://doi.org/10.1016/S1389-1286(99)00052-3.

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, et Robert E. Gruber. Bigtable : A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2) : 4 :1–4 :26, Juin 2008. ISSN 0734-2071. URL http://doi.acm.org/10.1145/1365815.1365816.

Gaozhao Chen, Shaochun Wu, Rongrong Gu, Yongquan Xu, Lingyu Xu, Yunwen Ge, et Cuicui Song. Data prefetching for scientific workflow based on hadoop. Dans Roger Lee, éditeur, *Computer and Information Science 2012*, pages 81–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-30454-5. URL https://doi.org/10.1007/978-3-642-30454-5_6.

Dazhao Cheng, Jia Rao, Changjun Jiang, et Xiaobo Zhou. Resource and deadline-aware job scheduling in dynamic hadoop clusters. Dans *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium*, IPDPS '15, pages 956–965,

Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4799-8649-1. URL http://dx.doi.org/10.1109/IPDPS.2015.36.

Crochford. Rfc 4627-the application/json media type for javascript object notation (json). http://tools.ietf.org/html/rfc4627, 2006.

Xiangming Dai et Brahim Bensaou. Scheduling for response time in hadoop mapreduce. *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016.

Jeffrey Dean et Sanjay Ghemawat. Mapreduce : Simplified data processing on large clusters. *Commun. ACM*, 51(1) :107–113, Janvier 2008. ISSN 0001-0782. URL http://doi.acm.org/10.1145/1327452.1327492.

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, et Werner Vogels. Dynamo : Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6) :205–220, Octobre 2007. ISSN 0163-5980. URL http://doi.acm.org/10.1145/1323293.1294281.

Ashley DeVan. The 7 v's of big data, 2016. URL https://impact.com/marketing-intelligence/7-vs-big-data/.

Rini Mary Nithila I Vinothini M Divya S, Kanya Rajesh R. Big data analysis and its scheduling policy- hadoop. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 17 : 36–40, Avril 2015. ISSN 1063-6692.

Chuntao Dong, Qingni Shen, Lijing Cheng, Yahui Yang, et Zhonghai Wu. Dans Kwok-Yan Lam, Chi-Hung Chi, et Sihan Qing, éditeurs, *Information and Communications Security*, pages 184–194, Cham, 2016. Springer International Publishing. ISBN 978-3-319-50011-9.

Park Dongchul et Kee Yang-Suk. In-storage computing for hadoop mapreduce framework : Challenges and possibilities. *IEEE TRANSACTIONS ON COMPUTERS*, 2015.

John Gantz et David Reinsel. Extracting value from chaos. pages 1–12, 01 2011.

John Gantz et David Reinsel. The digital universe in 2020 : Big data, bigger digital shadows, and biggest growth in the far east. Dans *IDC iView*, IDC Anal. Future. IDC, 2012.

Jyoti Gautam, Harshadkumar Prajapati, Vipul Dabhi, et Sanjay Chaudhary. A survey on job scheduling algorithms in big data processing. Dans *IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT'15)*, pages 1–11, 03 2015.

Blackett Gavin. Analytics network-o.r. analytics. http://www.theorsociety.com/Page/SpecialInterest/AnalyticsNetwork_analytics.aspx, 2013.

Song Ge, Meng Zide, Huet Fabrice, Magoules Frederic, Yu Lei, et Lin Xuelian. A hadoop mapreduce performance prediction method. *IEEE 10th International Conference on, Zhangjiajie*, pages 820–825, 2013.

Jimy Geetha, N UdayBhaskar, et P ChennaReddy. Data-local reduce task scheduling. 85 :598–605, 12 2016.

Sanjay Ghemawat, Howard Gobioff, et Shun-Tak Leung. The google file system. Dans *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43, 12 2003.

hadoop. Apache hadoop. http://Hadoop.apache.org.

HadoopTuto. Hadoop tutorial. http://developer.yahoo.com/hadoop/tutorial/module1.html.

Mohammad Hammoud, Suhail Rehman, et Majd Sakr. Center-of-gravity reduce task scheduling to lower mapreduce network traffic. *Cloud Computing (CLOUD), IEEE 5th International Conference on, Honolulu, HI*, pages 49–58, 2012.

Jiawei Han, Zhenhui Li, et Lu An Tang. Mining moving object, trajectory and traffic data. Dans Hiroyuki Kitagawa, Yoshiharu Ishikawa, Qing Li, et Chiemi Watanabe, éditeurs, *Database Systems for Advanced Applications*, pages 485–486, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-12098-5.

Ibrahim Hashem, Nor Anuar, Abdullah Gani, Ibrar Yaqoob, Feng Xia, et Samee Ullah Khan. Mapreduce : Review and open challenges. 109, 04 2016.

hbase. Hbase. http://hbase.apache.org/, 2013.

hdfs. Hadoop distributed file system. http://hadoop.apache.org/docs/r1.0.4/hdfsdesign.html.

Herodotou Herodotos. *Hadoop Performance Models*.

Herodotou Herodotos, Lim Harold, Luo Gang, Borisov Nedyalko, Dong Liang, Bilgen Fatma, et Shivnath Babu Cetin. Starfish : A self-tuning system for big data analytics. *in CIDR*, pages 261–272, 2011.

Han Hu, Yonggang Wen, Tat-Seng Chua, et Xuelong Li. Toward scalable systems for big data analytics : A technology tutorial. 2 :652–687, 01 2014.

Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, et Stephen J. Maybank. A survey on visual content-based video indexing and retrieval. 41 :797–819, 11 2011.

hypertable. Hypertable. http://hypertable.org/, 2013.

Shadi Ibrahim, Hai Jin, Lu Lu, Bingsheng He, Gabriel Antoniu, et Song Wu. Maestro : Replica-aware map scheduling for mapreduce. Dans *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID '12, pages 435–442, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4691-9. URL https://doi.org/10.1109/CCGrid.2012.122.

Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, et Dennis Fetterly. Dryad : Distributed data-parallel programs from sequential building blocks. Dans *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-636-3. URL http://doi.acm.org/10.1145/1272996.1273005.

Zhen Jia, Runlin Zhou, Chunge Zhu, Lei Wang, Wanling Gao, Yingjie Shi, Jianfeng Zhan, et Lixin Zhang. The implications of diverse applications and scalable data sets in benchmarking big data systems. Dans *Revised Selected Papers of the First Workshop on Specifying Big Data Benchmarks - Volume 8163*, pages 44–59, New York, NY, USA, 2014. Springer-Verlag New York, Inc. ISBN 978-3-642-53973-2. URL http://dx.doi.org/10.1007/978-3-642-53974-9_5.

HaiLong Wang HongFang Pan JinHong Cheng Xiao Qin Jiong Xie, FanJun Meng. Research on scheduling scheme for hadoop clusters. *Procedia Computer Science*, 18 : 2468–2471, 2013.

Gantz John et Reinsel David. The digital universe decade-are you ready. Dans *in Proc. White Paper, IDC*, 2010.

Layton Julia. How amazon works. http://knowwpcarey.com/article.cfm?aid=1171, 2013.

Henry F. Korth et Abraham Silberschatz. *Database System Concepts*. McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0-07-044752-7.

Kosmosfs. Kosmosfs. https://code.google.com/p/kosmosfs/.

Avinash Lakshman et Prashant Malik. Cassandra : A structured storage system on a p2p network. Dans *Proceedings of the Twenty-first Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '09, pages 47–47, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-606-9. URL http://doi.acm.org/10.1145/1583991.1584009.

Eugene L. Lawler, Jan Karel Lenstra, Alexander H.G. Rinnooy Kan, et David B. Shmoys. Sequencing and scheduling : Algorithms and complexity. Dans *Logistics of Production and Inventory*, volume 4 de *Handbooks in Operations Research and Management Science*, pages 445 – 522. Elsevier, 1993. URL http://www.sciencedirect.com/science/article/pii/S0927050705801896.

Ming Chang Lee, Jia Chun Lin, et Ramin Yahyapour. Hybrid job-driven scheduling for virtual mapreduce clusters. *IEEE Transactions on Parallel and Distributed Systems*, 27(6) :1687–1699, 2016.

Jinglun Li, Shengfei Shi, et Hongzhi Wang. Optimization analysis of hadoop. Dans Wanxiang Che, Qilong Han, Hongzhi Wang, Weipeng Jing, Shaoliang Peng, Junyu Lin, Guanglu Sun, Xianhua Song, Hongtao Song, et Zeguang Lu, éditeurs, *Social Computing*, pages 520–532, Singapore, 2016. Springer Singapore. ISBN 978-981-10-2053-7.

Jia-Chun Lin, Ingrid Chieh Yu, Einar Broch Johnsen, et Ming-Chang Lee. Abs-yarn : A formal framework for modeling hadoop yarn clusters. Dans Perdita Stevens et Andrzej Wąsowski, éditeurs, *Fundamental Approaches to Software Engineering*, pages 49–65, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

X. Lin, Z. Meng, C. Xu, et M. Wang. A practical performance model for hadoop mapreduce. *in CLUSTER Workshops*, pages 231–239, 2012.

Jun Liu, Tianshu Wu, Ming Wei Lin, et Shuyu Chen. An efficient job scheduling for mapreduce clusters. *International Journal of Future Generation Communication and Networking*, 8(2) :391–398, 2015.

Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, et Joseph M. Hellerstein. Distributed graphlab : A framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.*, 5(8) :716–727, Avril 2012. ISSN 2150-8097. URL https://doi.org/10.14778/2212351.2212354.

Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, et Grzegorz Czajkowski. Pregel : A system for large-scale graph processing. Dans *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0032-2. URL http://doi.acm.org/10.1145/1807167.1807184.

Pastorelli Mario, Carra Damiano, Dell'Amico Matteo, et Michiardi Pietro. Hfsp : Bringing size-based scheduling to hadoop. *IEEE Transactions on Cloud Computing*, 5(1) : 43–56, Jan.-March 2017. ISSN 2168-7161. URL doi.ieeecomputersociety.org/10.1109/TCC.2015.2396056.

Mohamed Merabet, Sidi mohamed Benslimane, Mahmoud Barhamgi, et Christine Bonnet. A predictive map task scheduler for optimizing data locality in mapreduce clusters. *International Journal of Grid and High Performance Computing (IJGHPC)*, 10(4) :1938–0259, 2018.

mongoDb. Mongodb. http://www.mongodb.org/, 2013.

Tatbul Nesime. Streaming data integration : Challenges and opportunities. Dans *in Proc. IEEE 26th Int. Conf. Data Eng. Workshops (ICDEW)*, page 155–158, 2010.

Leonardo Neumeyer, Bruce Robbins, Anish Nair, et Anand Kesari. S4 : Distributed stream computing platform. Dans *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 170–177, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4257-7. URL https://doi.org/10.1109/ICDMW.2010.172.

Dongchul PARK, Biplob DEBNATH, et David H.C. DU. A dynamic switching flash translation layer based on page-level mapping. *IEICE Transactions on Information and Systems*, E99.D(6) :1502–1511, 2016.

Russom Philip. Big data analytics. Dans *TDWI best practices report*. The Data Warehousing Institute (TDWI) Research, 2011.

Bryant Randal E. Data-intensive scalable computing for scientific applications. *Computing in Science & Engineering*, 13 :25–33, 07 2011. ISSN 1521-9615. URL doi.ieeecomputersociety.org/10.1109/MCSE.2011.73.

Pal Sankar K., Talwar Varun, et Mitra Pabitra. Web mining in soft computing framework : Relevance, state of the art and future directions. *IEEE Trans. Neural Netw*, 13 : 1163–1177, 2002.

Shaikh Sarah. Yarn versus mapreduce — a comparative study. Dans *Deepali Vora 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, page 1294–1297, 2016.

sdss. Sdss. http://www.sdss.org/.

Cameron Seay, Rajeev Agrawal, Anirudh Kadadi, et Yannick Barel. Using hadoop on the mainframe : A big solution for the challenges of big data. pages 765–769, 05 2015.

Yanling Shao, Chunlin Li, Wenyong Dong, et Liu Yunchang. Energy-aware dynamic resource allocation on hadoop yarn cluster. Dans *IEEE 18th International Conference on High Performance Computing and Communications ; IEEE 14th International Conference on Smart City ; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 364–371. IEEE, 2016.

Bing Shi et Ankur Srivastava. Thermal and power-aware task scheduling for hadoop based storage centric datacenters. Dans *International Green Computing Conference 2010, Chicago, IL, USA, 15-18 August 2010*, pages 73–83, 2010. URL https://doi.org/10.1109/GREENCOMP.2010.5598262.

Xuanhua Shi, Ming Chen, Ligang He, Xu Xie, Lu Lu, Hai Jin, Yong Chen, et Song Wu. Mammoth : Gearing hadoop towards memory-intensive mapreduce applications. 1, 07 2014.

Rohan Chakravarthy Shyam Deshmukh, Dr. J. V. Aghav. Job classification for mapreduce scheduler in heterogeneous environment. Dans *International Conference on Cloud, Ubiquitous Computing and Emerging Technologies*, 2013.

Statista. Number of mobile phone users worldwide from 2015 to 2020. https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/, 2018.

storm. Storm. http://storm-project.net/, 2013.

Mingming Sun, Hang Zhuang, Chang longLi, Kun Lu, et Xuehai Zhou. Hpso : Prefetching based scheduling to improve data locality for mapreduce clusters. *Algorithms and Architectures for Parallel Processing : 14th International Conference, ICA3PP, Dalian, China*, pages 82–95, 2014.

Mingming Sun, Hang Zhuang, Chang longLi, Kun Lu, et Xuehai Zhou. Scheduling algorithm based on prefetching in mapreduce clusters. *Applied Soft Computing*, 38 : 1109–1118, 2016.

S. Suresh et N.P. Gopalan. An optimal task selection scheme for hadoop scheduling. *IERI Procedia*, 10 :70 – 75, 2014. ISSN 2212-6678. URL http://www.sciencedirect.com/science/article/pii/S2212667814001415. International Conference on Future Information Engineering (FIE 2014).

Jian Tan, Xiaoqiao Meng, et Li Zhang. Coupling scheduler for mapreduce/hadoop. Dans *Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, pages 129–130, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0805-2. URL http://doi.acm.org/10.1145/2287076.2287097.

Bressoud Thomas et Tang Qiuyi. Results of a model for hadoop yarn mapreduce tasks. Dans *IEEE International Conference on Cluster Computing*, pages 443–446, 2016.

Nidhi Tiwari, Santonu Sarkar, Umesh Bellur, et Maria Indrawan. Classification framework of mapreduce scheduling algorithms. *ACM Comput. Surv.*, 47(3) :49 :1–49 :38, Avril 2015. ISSN 0360-0300. URL http://doi.acm.org/10.1145/2693315.

Troppens Ulf, Müller-Friedt Wolfgang, Erkens Rainer, et Haustein Nils. *Networks Explained : Basics and Application of Fibre Channel SAN, NAS, ISCSI, Infiniband and FCoE*. Wiley, New York, NY, USA, 2011.

Mohd Usama, Liu Mengchen, et Chen Min. Job schedulers for big data processing in hadoop environment : testing real-life schedulers using benchmark programs. *Digital Communications and Networks*, 3(4) :260 – 273, 2017. ISSN 2352-8648. URL http://www.sciencedirect.com/science/article/pii/S2352864817301955. Big Data Security and Privacy.

Wil van der Aalst. Process mining : Overview and opportunities. *ACM Trans. Manage. Inf. Syst.*, 3(2) :7 :1–7 :17, Juillet 2012. ISSN 2158-656X. URL http://doi.acm.org/10.1145/2229156.2229157.

Deepak Vohra. *Practical Hadoop Ecosystem : A Definitive Guide to Hadoop-Related Frameworks and Tools*. Apress, Berkely, CA, USA, 1st édition, 2016. ISBN 1484221982, 9781484221983.

Guanying Wang, Aleksandr Khasymski, Krish K. R., et Ali Butt. Towards improving mapreduce task scheduling using online simulation based predictions. *Parallel and Distributed Systems (ICPADS), International Conference on, Seoul*, pages 299–306, 2013.

Wang Weina et Ying Lei. Data locality in mapreduce : A network perspective, performance evaluation. 96 :1–11, 2016.

Qiaomin Xie, Mayank Pundir, Yi Lu, Cristina L. Abad, et Roy H. Campbell. Pandas : Robust locality-aware scheduling with stochastic delay optimality. *IEEE/ACM Trans. Netw.*, 25(2) :662–675, Avril 2017. ISSN 1063-6692. URL https://doi.org/10.1109/TNET.2016.2606900.

Bin Xu, Jiajun Bu, Chun Chen, et Deng Cai. An exploration of improving collaborative recommender systems via user-item subgroups. Dans *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 21–30, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5. URL http://doi.acm.org/10.1145/2187836.2187840.

Liu Yang, Zeng Yukun, et Piao Xuefeng. High-responsive scheduling with mapreduce performance prediction on hadoop yarn. *2016 IEEE 22nd International Conference on*

Embedded and Real-Time Computing Systems and Applications (RTCSA), pages 238–247, 2016.

Dongjin Yoo et Kwang Sim. A comparative review of job scheduling for mapreduce. Dans *Cloud Computing and Intel. Syst. (CCIS), IEEE Int*, pages 203–210. IEEE, 2011.

Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, et Ion Stoica. Delay scheduling : A simple technique for achieving locality and fairness in cluster scheduling. Dans *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 265–278, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-577-2. URL http://doi.acm.org/10.1145/1755913.1755940.

Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, et Ion Stoica. Improving mapreduce performance in heterogeneous environments. Dans *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855741.1855744.

Jianhong Zhai, Hongli Zhang, Xiaorou Zhong, Wei Li, Lai Wang, et Zeyu He. Energy-efficient hadoop green scheduler. *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, pages 335–340, 2016.

Xiaohong Zhang, Zhiyong Zhong, Shengzhong Feng, Bibo Tu, et Jianping Fan. Improving data locality of mapreduce by scheduling in homogeneous computing environments. Dans *Proceedings of the 2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications*, ISPA '11, pages 120–126, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4428-1. URL http://dx.doi.org/10.1109/ISPA.2011.14.

Yanming Shen Zhigang Wang. Job-aware scheduling for big data processing. Dans *International Conference on Cloud Computing and Big Data*. IEEE, 2015.

جدولة فعالة للمهام لتطبيقات Big Data

ملخص

يظهر Hadoop كإطار مهم لمعالجة البرامج المتوازية مع نطاق واسع من البيانات. يعد تحسين موقع البيانات طريقة فعالة لتسريع هذه البرامج المتوازية. في هذه الأطروحة ، نوفر استراتيجية جدولة تنبؤية والتي تعمل على جلب بيانات المدخلات من مهام Map إلى Reduce قبل بداية المعالجة الحقيقية. نحن نستخدم نموذج الشبكة العصبية المبنية على التوقع والبيانات لتحديد المواقع لجدولة المهام. تظهر نتائج التقييمات أن برنامج جدولة المهام التنبؤي Map يمكن أن يعمل على تحسين كل من موقع البيانات ومدة تنفيذ مهام Map بشكل كبير.

الكلمات المفتاحية تموقع البيانات ، Big Data ، جدولة مهام Map ، MapReduce ، التحميل المسبق.

Un ordonnancement efficace des tâches pour les applications Big data

Résumé

Hadoop apparaît comme un framework important pour le traitement de programmes parallèles contenant des données à grande échelle. L'amélioration de la localisation des données est un moyen efficace d'accélérer ces programmes parallèles. Dans cette thèse, nous fournissons une stratégie d'ordonnancement prédictif qui pré-charge les données d'entrée des tâches map aux nœuds avant le début du traitement. Nous utilisons un modèle de réseau de neurones pour la prédiction et un algorithme basé sur la localité de données pour l'ordonnancement des tâches. Les résultats des évaluations montrent que notre ordonnanceur prédictif de tâches map peut considérablement améliorer la localisation des données et la durée d'exécution des tâches map.

Mots Clés : localisation des données, Big Data, ordonnancement des tâches Map, MapReduce, préchargement.

Efficient task scheduling for Big Data applications

Abstract

Hadoop emerges as an important framework to process parallel programs with large scale of data. Improving data locality is an efficient way to speed up these parallel programs. In this thesis, we provide a predictive scheduling strategy which prefetch input data of map tasks to nodes before the real computation start. We use a Neural network model for prediction and data locality based algorithm for tasks scheduling. Evaluations results show that our Predictive Map Task Scheduler reduces can greatly improve both data locality and execution time of map tasks.

Keywords : Data Locality, Big Data, Map Task Scheduling, MapReduce, Prefetching.