REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED

FACULTÉ DES SCIENCES EXACTES
DEPARTEMENT D'INFORMATIQUE

Mémoire de fin d'études

pour l'obtention du diplôme de

MASTER EN INFORMATIQUE

Option: Systèmes Distribués et Intelligence Artificielle

Thème

Etude comparative des performances des SGBDs NoSQL

Cas d'étude : MongoDB Vs Cassandra

Réalisé par :
- BEDDA Ilias
- ABID Hadda

Encadré par:
- M. KHELAIFA Abdennacer

Année académique 2017/2018

Remerciements

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

من لم يشكر الناس لم يشكر الله

En second lieu, nous tenons à remercier notre encadreur M. Abdennacer Khelaifa, pour son précieux conseil et son aide durant toute la période du travail.

Nous tenons à exprimer nos sincères remerciements à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Nos remerciements vont aussi au corps administratif du Département de l'informatique

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicace

الحمد لله الذي قال و صدق "ادعوني استجب لكم"

أهدي هذا العمل البسيط إلى:

والديا الكريمين أطال الله في عمرهما

زوجتي التي كافحت و صبرت لأصل إلى هنا

أبنائي الذين اقتطعت من وقتي معهم، إخوتي و أصهاري الأعزاء

نبيل، حسام، رضا

كل من دعا لي

إليكم جميعا ... شكرا

إلياس

Je dédie ce modeste travail

A l'esprit de mes parents

Mon binôme Ilias Qui m'a beaucoup aidé dans ma carrière

Ma sœur Noura,

A tous ceux qui me soutiennent moralement.

Hadda Abid

Résumé

Les bases NoSQL sont de nouvelles bases de données qui ont vu le jour pour la plupart il y a moins de 15 ans permettant de traiter de très grands volumes de données informatiques et de combler les lacunes des bases de données classique. Ils sont utilisés par les grandes entreprises informatique avec les technologies Bigdata et cloud computing pour répondre à leurs besoins d'évolutivité.

Les bases de données NoSQL sont basées sur des différents modèles de données autres que le modèle relationnel, par exemple: clé-valeur, orienté colonne, orienté graphe, etc. le problème qui se pose dans ce contexte est le choix du meilleur SGBD pour une telle situation.

Le but de ce travail est d'étudier les différentes bases NoSQL disponibles sur le marché (architecture et modèle de données) et d'analyser les performances de deux d'entre elles, Cassandra et MongoDB en utilisant le célèbre outil de test: YCSB. La finalité est de connaître les performances de chacune des deux bases de données et d'effectuer une comparaison à partir des résultats des tests générés par YCSB.

Mots clés : NoSQL, Big Data, ACID, CAP, Cassandra, MongoDB, YCSB.

Abstract

NoSQL databases are a new databases, most of them have been established less than 15 years ago allowing the processing of a huge amount of IT data and filling the gaps of the classic databases. They are used by the big IT companies with Big Data and Cloud Computing technologies to meet their needs of scalability.

NoSQL databases are based on different data models other than the relational model, for example: key-value, column oriented, graph oriented, etc. the question to be asked within this context is: how to choose the best DBMS for such a situation?

The aim of this work is to study the different NoSQL databases available on the market (design and data model) and to analyze the performances of two of them, Cassandra and MongoDB, using the famous test tool: YCSB. The purpose is to know the performances of both of the databases and to perform a comparison based on the test results generated by YCSB.

Keywords:NoSQL, Big Data, ACID, CAP, Cassandra, MongoDB, YCSB.

ملخص

قواعد البيانات NoSQL هي قواعد بيانات جديدة و التي ظهر أغلبها في الخمسة عشر السنة الماضية تمكن من معالجة كميات هائلة من البيانات كما أنها تسد نقائص قواعد المغطيات الكلاسيكية. تستعمل قواعد البيانات NoSQL من طرف شركات الإعلام الآلي الكبرى مرفوقة بتكنولوجيات البيانات الضخمة "Big Data" و الحوسبة السحابية "Cloud Computing"، كل هذا من أجل الإستجابة لمتطلباتها في التوسع. المشكل المطروح هو كيفية اختيار المناسب لوضعية ما.

الهدف من هذه الدراسة هو دراسة مختلف قواعد البيانات NoSQL المتوفرة في السوق (من حيث الهندسة و نمط المعلومات) و تحليل قدرات اثنين منها، Cassandra و MongoDB، و ذلك باستعمال أداة الأختبار المعروفة YCSB.

الغرض من كل هذا هو معرفة قدرات و آداء كل من قاعدتي البيانات و إجراء مقارنة بينهما انطلاقا من نتائج الإختبارات المتحصل عليها من YCSB.

Sommaire

Liste des figures

Liste des abréviations

YCSB : Yahoo! Cloud Serving Benchmark

SGBD : Système de Gestion de Bases de Données

ACID : Atomicité, Cohérence, Isolation, Durabilité

BASE : Basically Available Soft-state Eventually consistent

BD : Base de Données

LDD : Langage de Définition de Données

LMD : Langage de Manipulation de Données

SGBDR : Système de Gestion de Base de Données Relationnelles

CAP : Consistency, Availability, Partition tolerance

BDR : Base de données Relationnelle

XML: eXtensible Markup Language

E/S : Entrée/Sortie

CRUD: Create, Read, Update, Delete

BSON: Binary JSON

CQL : Cassandra Query Language

IaaS: Infrastructure as Service

PaaS: Platform as Service

SaaS: Software as Service

VM: Virtual Machine

CCM: Cassandra Cluster Manager

GPS: Global Positioning System

CPU: Central Process Unit

RAM: Random Access Memory

RF: Replication Factor

Introduction générale

Depuis le début du 21ème siècle, l'internet est devenu accessible à des milliards d'utilisateurs, et depuis, un nouveau terme fut connu en informatique, les bases de données non relationnelles, ou ce qu'est communément appelé le NoSQL. De plus en plus beaucoup de grandes compagnies manipulent des grands volumes de données non structurées ont adopté le NoSQL. Par conséquent, de nombreux développeurs ont commencé de s'orienter vers ce nouveau produit et d'offrir des solutions adéquates et futuristes.

Les bases de données relationnelles ont né dans une époque des ordinateurs personnels et les applications de gestion, avant même l'arrivée de l'internet, le Cloud, le Big Data, les smartphones, … ces bases de données relationnelles telles que l'Oracle ont été conçu pour travailler sur un seul serveur puissant. Pour faire accroitre la capacité d'une base de données, la seule solution est d'augmenter les capacités du processeur, de la RAM ou du disque dur. Le problème que cette augmentation a un seuil, une limite !

L'objectif de ce travail consiste à définir le terme NoSQL, pourquoi il s'est existé, expliquer l'architecture de ses fameuses bases disponibles sur le marché et quelles sont ses attentes.

Nous l'aborderons d'un point de vue historique en jetant la lumière sur les bases de données relationnelles classiques et leurs limites qui ont mené à l'apparition du NoSQL. Nous passerons ensuite aux définitions des principaux concepts du NoSQL et explorerons les différents types de ces bases de données les plus connus; leurs avantages et leurs lacunes.

Enfin, une étude expérimentale comparative de deux base de données NoSQL, à savoir ; le Mongo DB et le Cassandra, et leurs performances avec des paramètres technique, pour arriver à une analyse de performance utile en utilisant YCSB, un outil très connu pour sa puissance de test et utilisé dans plusieurs travaux d'évaluation des bases de données NoSQL.

Nous pensons que ce travail peut servir de support aux lecteurs et spécialement les étudiants en informatique et les aider à avoir une initiation «Qu'est-ce que le NoSQL?» et «Pourquoi le NoSQL?»

Chapitre I : Les bases de données classiques

1. Introduction

De nos jours, nous stockons presque toutes nos informations sur des supports numériques, nos photos de famille et professionnelles, nos données des réseaux sociaux, nos musiques, nos vidéos, nos documents scannés ; le volume de données que nous produisons et consommons a pris une ampleur exponentielle.

Les bases de données sont devenues incontournable dans tous les domaines, l'informatique, l'astronomie, les sciences naturelles, la cybercriminalité, la biochimie… etc. Il n y a pas si longtemps, les bases de données relationnelles dominaient le domaine de la gestion de données. Nous abordons dans ce chapitre la définition des bases de données relationnelles et leurs limites qui ont mené à l'apparition d'un nouveau système de stockage de données.

2. Bases de données BD

Littéralement, une BD signifie «ni rien qu'un ensemble *organisé d'informations qui peut exister au cours d'une longue période de temps, souvent, plusieurs années* » (5)

Scientifiquement – en informatique –, une base de données est une collection de données gérée par un Système de Gestion de Bases De Données SGBD.

3. Système de Gestion de Bases de Données SGBD :

Un SGBD assure la création de la base de données, la manipulation d'une grande quantité de données avec efficacité, en leur permettant de persister pour une longue période de temps.

C'est un ensemble de routines qui permet de:

- Créer une base de données et de spécifier son schéma (structure logique) en utilisant un Langage de Définition de Données «LDD».
- Manipuler et traiter et modifier les données via des requêtes, en utilisant un langage de manipulation de données «LMD».
- Maintenir la base via des routines de gestion, d'administration et de contrôle.
- Stocker d'énormes volumes de données pendant une longue période de temps.
- Garantir une durabilité et une récupération des données en cas de panne ou de défaillance.

Tout en assurant l'accès, la sécurité, la cohérence, la confidentialité, l'intégrité et l'évolution des données à plusieurs utilisateurs simultanément.

4. Historique

Le 1er système de gestion de fichier est apparu vers la fin des années 60 du 20ème siècle, il était l'évolution du système classique de fichiers qui assure le stockage d'un ensemble de données sur une durée de temps et permet lui aussi le stockage d'un grand volume de données ; mais par contre, il ne garantit pas le recouvrement en cas de perte ni la liaison efficace entre plusieurs fichiers. En plus, le système de fichier ne fournit pas un LMD direct pour interroger les données. Son support au schéma se limite à la création d'une arborescence pour les fichiers. Et malgré qu'il permette l'accès simultané à un fichier en lecture, il ne permet pas la mise à jour simultanée.

En 1970, Ted Codd (6), mathématicien et chercheur à IBM[1], a publié un article intitulé « A Relational Model of Data for Large Shared Data Banks » ou il a proposé l'organisation des données dans des tables appelées relations; un modèle fondé sur la théorie mathématique des ensembles et sur la notion de base qui lui est rattachée, i.e. la relation.

	SupplierID	CompanyName	ContactName	ContactTitle	Address	City	Re
1	1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	49 Gilbert St.	London	N
2	2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	P.O. Box 78934	New Orleans	L
3	3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	707 Oxford Rd.	Ann Arbor	M
4	4	Tokyo Traders	Yoshi Nagase	Marketing Manager	9-8 Sekimai Musashino-shi	Tokyo	N
5	5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra	Export Administrator	Calle del Rosal 4	Oviedo	A
6	6	Mayumi's	Mayumi Ohno	Marketing Representative	92 Setsuko Chuo-ku	Osaka	N
7	7	Pavlova, Ltd.	Ian Devling	Marketing Manager	74 Rose St. Moonie Ponds	Melbourne	Vi
8	8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	29 King's Way	Manchester	N
9	9	PB Knäckebröd AB	Lars Peterson	Sales Agent	Kaloadagatan 13	Göteborg	N
10	10	Refrescos Americanas LTDA	Carlos Diaz	Marketing Manager	Av. das Americanas 12.890	Sao Paulo	N

Figure 1- Une table d'une BD relationnelle

Une BD relationnelle est un ensemble de tables contenant des données ajustées sur des catégories prédéfinies. Chaque table contient une ou plusieurs catégories comme colonne, et chaque ligne contient une instance unique des catégories définis par les colonnes. Les utilisateurs peuvent accéder ou interroger les données de différentes façons sans avoir réorganiser les tables de la BD.

[1] IBM : International Business Machines Corporation

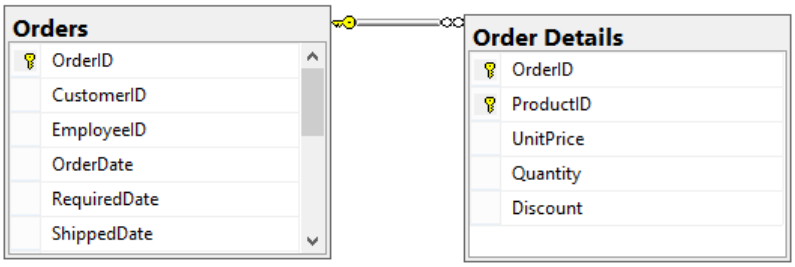

Figure 2- Tables et relation

Pendant des décennies, les BD relationnelles ont été utilisée pour stocker les données relationnelles.

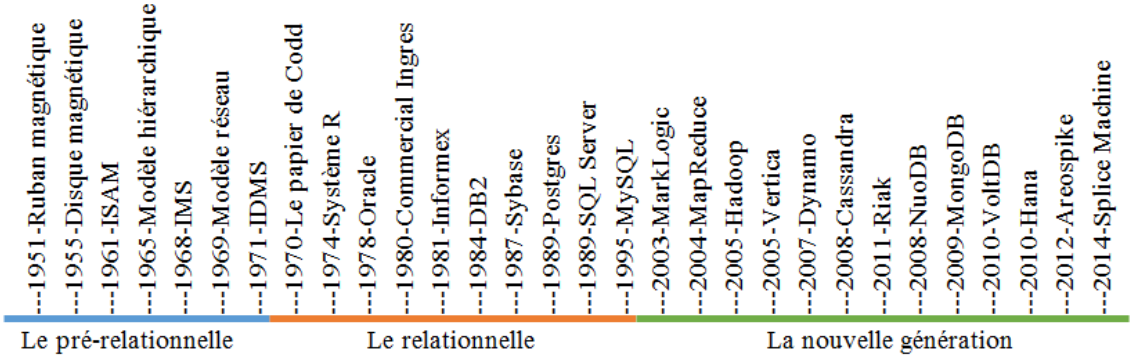

Figure 3- La chronologie d'apparition des bases de données les plus connus

5. ACID

On appelle une transaction, un ensemble d'opérations unitaires indissociables qui doivent réussir ou échouer ensemble.

Le principe de base des SGBD relationnels est les propriétés ACID des transactions ; chaque transaction doit garantir les propriétés suivantes :

- Atomicité : Fonctionnement en « tout ou rien », dans une opération bancaire, un transfert de fond doit être fait comme une transaction « toute ou rien » transaction. Tous les systèmes adoptant l'atomicité doivent prendre en considération les cas de défaillance et de panne tels que le crash des disques, la défaillance du réseau, du matériel ou simplement les erreurs de programmation.
- Cohérence : Données cohérentes avant & après chaque transaction. Dans la même opération suscitée, lors du transfert de fonds, la balance ne doit pas être changée. C'est le principe de la cohérence. Cela veut dire, qu'il ne faut jamais avoir un compte débité alors que le deuxième n'est pas encore créditer.

C'est la responsabilité du SGBD de bloquer tout rapport durant une opération atomique. Cela a son impact sur la vitesse du système quand plusieurs transactions atomiques s'exécutent simultanément.

- Isolation : fait référence au concept que chaque action intermédiaire d'une transaction est invisible à l'autre. Par exemple, la transaction qui alimente un compte bancaire est indépendante de celle qui débite le compte.
- Durabilité : fait référence au concept qu'une fois une transaction validée, elle ne sera jamais remise en cause ; elle est permanente. Par exemple, si le système bancaire crash dans la nuit, et ils devront le récupérer à partir d'une copie de backup, il doit y avoir un journal de transactions dans un endroit séparé et qui assure la réexécution de toutes les transactions après la récupération des données.

Toutefois, ces propriétés ne sont pas applicables ensemble dans un contexte distribué.

Les SGBDR les plus fameux qui ont dominé depuis les années 80 sont Oracle, MySQL et SQL Server.

6. Limitations des SGBDRs

- Toute la structure d'une table, les noms distincts des champs et leurs types uniques, doit être créé avant l'insertion de la première ligne.
- L'évolutivité, le passage à l'échelle: est la capacité d'un système de s'adapter avec l'augmentation de la charge des données et des ressources. "*les BD relationnelles ne fonctionnent pas parfaitement dans un environnement distribué à cause de la jonction difficile des tables distribuées*"[2] ; "*elles n'étaient pas conçues pour fonctionner avec des données partitionnées, alors leur distribution est une corvée*"[3]

  Normalement, pour augmenter la performance, on doit avoir des machines plus puissantes dotées de grands volumes de stockage, ce qui est le passage vertical. L'autre solution est le passage horizontal ; distribuer la puissance et le volume sur

[2] Jeremy Zawodny, Ingénieur software au magazine Craigslist
[3] Stephen O'Grady, analyste chez RedMonk, société de recherches du marché

des machines normales, ceci est pratique pour des sites web, mais les bases de données sont gelées par leur schémas.

- La complexité: toutes les données doivent être incluses dans des tables, respectant des contraintes d'intégrité, qu'est-ce qui rend la structure de la BD difficile et l'interrogation lente.
- SQL: "*SQL est conçu pour travailler avec des données structurées; des tables avec des informations fixes, mais il est difficile de l'utiliser avec d'autres types de données. SQL peut contenir plusieurs lignes de code complexe et ne fonctionne pas bien avec le développement souple moderne* "[4].
- La cohérence : parce qu'elle est l'un des facteurs majeures des SGBD relationnels, le passage à l'échelle horizontal est un vrai défi, si ce n'est pas impossible.
- Malgré leurs qualités, les SGBD relationnels ne sont pas compatibles avec la disponibilité et la performance qu'exige l'extensibilité des applications web. MySQL et PostgressSQL ne peuvent pas gérer une base de données distribuée sur deux serveurs actifs, alors qu'Oracle peut le faire, mais l'opération coûte cher pour avoir les serveurs, un espace de stockage partagé et plusieurs licences, elle risque de ne pas être rentable.
    - La difficulté de suivre le rythme d'évolution du data Waterhouse, du Grid,
    - du Web 2.0 et du Cloud.

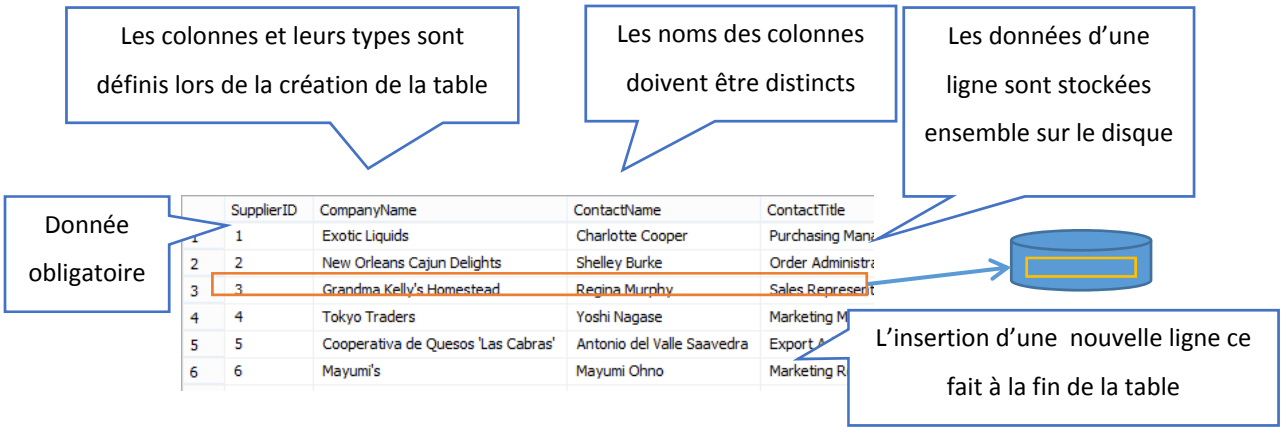

Figure 4 - La rigidité d'une table relationnelle

[4] Stefan Edlich, Pr à l'université des sciences appliquées de Beuth à Berlin

7. Conclusion

Les technologies de bases de données relationnelles et transactionnelles, qu'on pourrait nommer par "technologies SQL", sont devenues au fil du temps une référence absolue pour l'industrie des bases de données.

On peut considérer SQL comme un standard de fait pour toute problématique ayant trait au stockage et à la manipulation de données.

Pourtant, un certain nombre de limitations importantes sont apparues au fil des années et l'absolu est devenu obsolète!

Dans le scénario de l'Internet actuel, où d'importants volumes de données structurées de façon variable doivent être traités rapidement, de nouvelles méthodes ont été élaborées pour pallier les insuffisances des BD SQL et surtout en termes d'évolutivité.

Chapitre II : Le NoSQL

1. Introduction

Nous parlerons dans ce chapitre sur le théorème CAP qui a été à la base des recherches qui ont mené à l'apparition du NoSQL ; puis la notion du BASE, le dérivé du théorème. Nous définirons quelques notions liées, telle que le Big Data et le Cloud.

2. Le tournant, le théorème CAP

Lors du symposium sur les principes de l'informatique distribué (7), organisé en 2000 à l'université de Berkeley en Californie, Eric Brewer a donné une présentation sur son expérience sur les changements des bases des données distribuées, et par laquelle, il a présenté pour la première fois son théorème CAP.

Le principe de ce théorème est le suivant : toute base de données distribuée doit théoriquement respecter trois contraintes:

- La cohérence (**C**onsistency) : Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas.
- La disponibilité (**A**vailability) : Une donnée doit être toujours disponible, quel que soit l'état du système.
- La tolérance à la distribution (**P**artition Tolérance) : Une requête doit fournir un résultat correct quel que soit le nombre de serveurs.

Alors que d'après le théorème de Brewer, pratiquement, à un instant donné, que deux contraintes à la fois peuvent être garanties par une base de données.

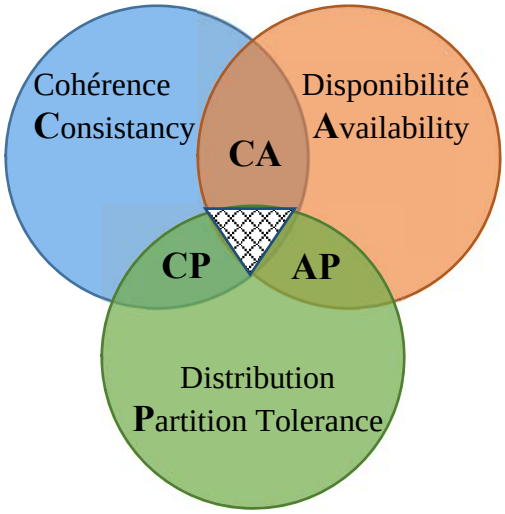

Figure 5- Les contraintes du NoSQL

Les trois options qui peuvent exister sont :

1. Marginaliser la tolérance à la distribution (CA) : le système ne prend pas en considération la distribution des données sur un réseau. C'est le cas typique des SGBDRs.

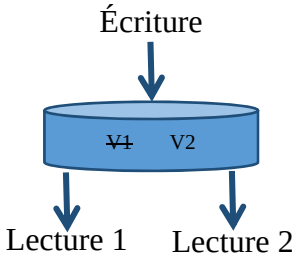

Figure 6- CA Cohérence + Disponibilité

Dans la Figure 6, les deux requêtes de lecture concurrente sur une même donnée, retournent le même nouveau résultat et sans délai d'attente.

2. Marginaliser la cohérence (AP): Dans le cas de la distribution, les données peuvent être sollicitées, mais à cause de la rupture des nœuds, la cohérence n'est pas garantie, parce que les mises à jour sont asynchrones sur le réseau. Cette option s'intéresse à fournir un temps de réponse rapide.

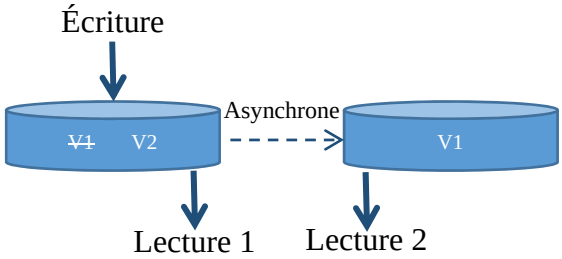

Figure 7- AP Disponibilité + Distribution

Dans la Figure 7, la lecture1 retourne v2 alors que la lecture2 retourne v1. Cassandra utilise cette option, avec des temps de réponse très appréciables mais avec des résultats non garantis à 100%.

3. Marginaliser la disponibilité (CP) : Les données ne peuvent être utilisées que si leur cohérence est garantie. Une donnée mise à jour sur un nœud, doit être bloquée sur les autres nœuds jusqu'à la propagation de la nouvelle version sur tout le réseau. Dans un environnement distribué, une base de données prend un temps considérable pour avoir un état cohérent, ce qui rend la disponibilité relative.

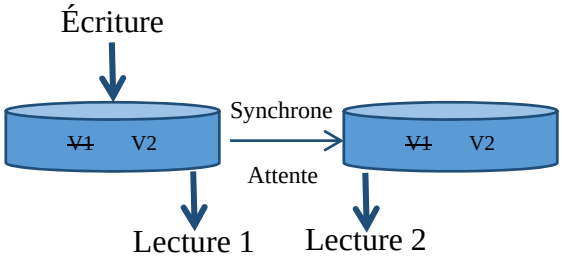

Figure 8- CP Cohérence + Distribution

Dans la Figure 8 : Les requêtes Lecture1 et Lecture2 attendent la synchronisation pour avoir le résultat v2. Le résultat est cohérent mais avec un délai de latence. MongoDB utilise cette option des BD NoSQL.

L'option de marginaliser la distribution n'est pas réaliste, car de nos jours, il n'est pas pratique, voire inimaginable de travailler dans un environnement non distribué.

3. BASE

D'après la conclusion du paragraphe précèdent, il reste le choix entre la cohérence représentée par ACID et la disponibilité représentée par BASE (Basically Available Soft-state Eventually consistent), le terme inventé par Brewer et ses successeurs pour désigner le cas d'une cohérence éventuelle.

- Basically Available (la disponibilité): Une réponse est garantie pour chaque requête, que ce soit avec succès ou avec échec, quelle que soit la charge de la base de données.
- Soft-state : L'état du système peut se changer lors des mises à jour, la base n'est pas forcement cohérente à tout instant.
- Eventually consistent : La base de données peut être incohérente temporairement mais elle doit être cohérente à un terme.

D'après Brewer, le choix n'est pas binaire, on peut avoir un mélange proportionnel des niveaux de cohérence et de disponibilité pour avoir de meilleurs résultats.

En 2005, Google fut le plus grand site sur le web, et les SGBDs relationnels ont montré une insuffisance flagrante pour faire face à la montée de la vitesse et aux volumes des données. Les défis de Big Data que les entreprises font face aujourd'hui, Google les a déjà rencontrés il y a presque 20 ans. (8).

4. Big Data

« *Le volume de données stockées sur les ordinateurs est estimé à 300 exabyte (300 billion gigabyte), ce chiffre augmente 28% chaque année, ce qui a donné naissance à une nouvelle génération de ressources de stockage de données ; leBig Data.* » (9)

Le BigData désigne « *des ensembles de données devenus si volumineux qu'ils dépassent l'intuition et les capacités humaines d'analyse et même celles des outils informatiques classiques de gestion de base de données ou de l'information* » (10).

« *Il est caractérisé par les 3 V : le Volume, une grande quantité de données ; la Variété, différents types de données, et la Vitesse, l'accumulation des données.* » (9)

« Big Data = Transactions + Interactions + Observations » (11)

La figure suivante illustre cette définition :

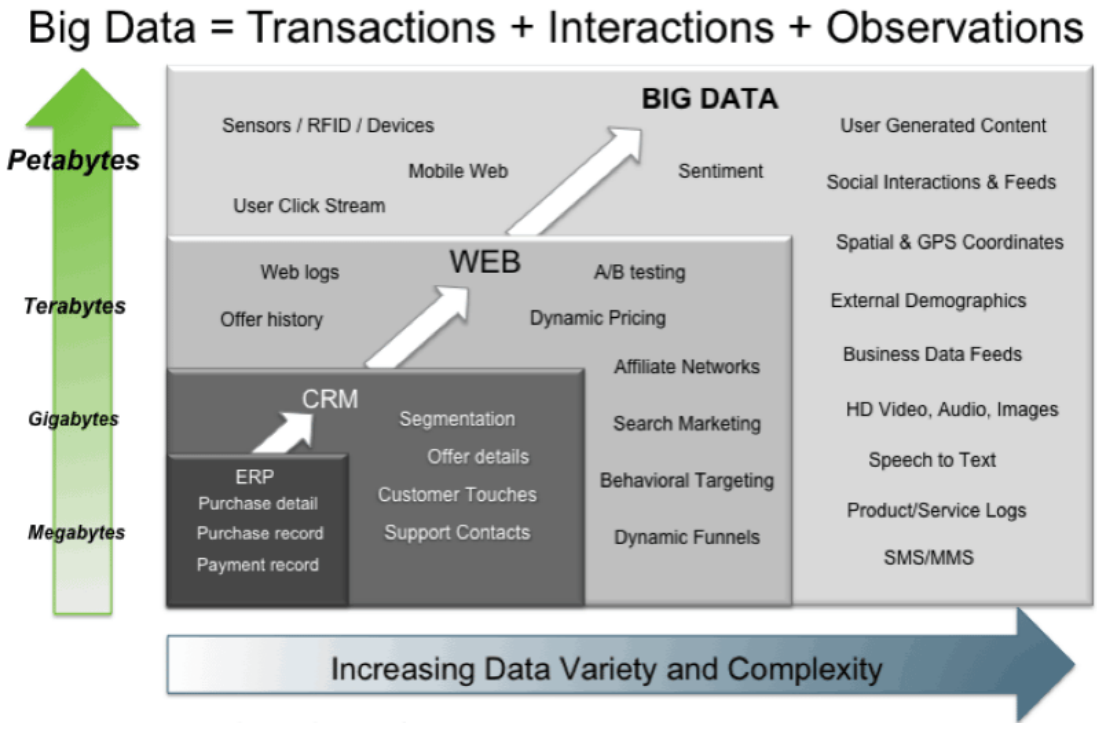

Figure 9-Transaction Big Data avec les interactions et les observations (11)

Les données hautement structurées telles que ERP (Enterprise Resource Planning) ou CRM (Customer Relationship Management) sont stockées dans des bases de données SQL classiques. Elles sont basées essentiellement sur la notion des transactions.

Une interaction est un échange entre humain et machine ; les journaux du web, les clicks des utilisateurs, les interactions et les flux sociaux sont tous des interactions de données.

Les observations sont une notion de l' l'« internet of things », toutes les données issues des capteurs de chaleur, de pression, des coordonnées GPS sont des sources de données d'observation.

Toutes ces données ont poussé les SGBDR à leurs limites, ce qui a mené au développement des bases de données distribuées, évolutives horizontale et non relationnelles.

En 2003, Google a révélé des détails sur sa structure de bases des données distribuées: le BigTable. Son implémentation était le HBase. Plus tard en Octobre 2007, Yahoo a lancé son produit, le Hadoop, qui est à la base le projet HBase le plus maturé.

Dans la même année 2007, le géant de la vente électronique, Amazon, publie un article qui annonçait la naissance officielle du NoSQL, il présentait son SGBD NoSQL Dynamo.

5. Cloud Computing

En termes simples, l'informatique en nuage (Cloud Computing) est la fourniture de services informatiques: serveurs, stockage, bases de données, mise en réseau, logiciels, analyses et plus via Internet.

6. L'utilisation

Aujourd'hui, vous utilisez probablement le Cloud Computing dès maintenant, même si vous ne le réalisez pas. Si vous utilisez un service en ligne pour envoyer des e-mails, modifier des documents, regarder des films ou des émissions télévisées, écouter de la musique, jouer à des jeux ou stocker des images et autres fichiers, vous êtes en train d'utiliser un service Cloud. Voici quelques exemples de ce que vous pouvez faire avec le Cloud:

- Créer de nouvelles applications et services
- Stocker, sauvegarder et récupérer des données
- Sites web et blogs hôtes
- Stream audio et vidéo
- Livrer des logiciels à la demande
- Analyser les données pour les modèles et faire des prédictions

7. Les types des services

Les services de Cloud se répartissent en trois grandes catégories: infrastructure en tant que service (IaaS), plate-forme en tant que service (PaaS) et logiciel en tant que service (Saas).

Infrastructure en tant que service (IaaS)

La catégorie la plus élémentaire des services de Cloud, avec IaaS, vous louez une infrastructure informatique, des serveurs et des machines virtuelles (VM), des systèmes de stockage, des réseaux et des systèmes d'exploitation, à partir d'un fournisseur de Cloud.

Plateforme en tant que service (PaaS)

La plate-forme en tant que service (PaaS) fait référence aux services de Cloud qui fournissent un environnement à la demande pour développer, tester, fournir et gérer des applications logicielles. Le PaaS est conçu pour faciliter la création rapide d'applications Web ou mobiles par les développeurs, sans se soucier de la configuration ou de la gestion de l'infrastructure sous-jacente des serveurs, du stockage, du réseau et des bases de données nécessaires au développement.

Logiciel en tant que service (SaaS)

Le logiciel en tant que service (SaaS) est une méthode permettant de fournir des applications logicielles sur Internet, à la demande et généralement sous forme d'abonnement. Avec SaaS, les fournisseurs de Cloud hébergent et gèrent l'application logicielle et l'infrastructure sous-jacente, et gèrent toute la maintenance, comme les mises à niveau logicielles et les correctifs de sécurité. Les utilisateurs se connectent à l'application via Internet, généralement avec un navigateur Web.

8. Les types de déploiement

Cloud public

Les Clouds publics sont détenus et exploités par un fournisseur de services Cloud tiers, qui fournit leurs ressources informatiques tels que les serveurs et le stockage via Internet. Microsoft Azure est un exemple de Cloud public. Avec un Cloud public, l'ensemble du matériel, logiciels et autres infrastructures de support sont détenus et gérés par le fournisseur de Cloud. Vous accédez à ces services et gérez votre compte à l'aide d'un navigateur Web.

Ce modèle est caractérisé par :

- Requiert de lourds investissements pour le fournisseur de services
- Offre un maximum de flexibilité.
- N'est pas sécurisé.

Cloud privé

Un Cloud privé fait référence aux ressources de Cloud utilisées exclusivement par une seule entreprise ou organisation. Un Cloud privé peut être physiquement situé sur le centre de données sur site de l'entreprise. Certaines entreprises paient également des fournisseurs de services tiers pour héberger leur Cloud privé. Un Cloud privé est un Cloud dans lequel les services et l'infrastructure sont gérés sur un réseau privé.

Cloud hybride

Les Clouds hybrides combinent des Clouds publics et privés, liés par une technologie permettant de partager des données et des applications entre eux. En permettant aux données et aux applications de se déplacer entre des Clouds privés et publics, le Cloud hybride offre aux entreprises une plus grande flexibilité et davantage d'options de déploiement.

9. NoSQL

NoSQL désigne une famille de systèmes de gestion de base de données (SGBD) qui s'écarte du paradigme classique des bases relationnelles. L'explicitation du terme la plus populaire de l'acronyme est « Not only SQL » en anglais, « pas seulement SQL ». Les données ne sont pas relationnelles et le langage SQL n'est pas utilisé pour la définition et la manipulation des données.

C'est l'impact du fait que tout le monde a commencé à se concentrer sur la création de bases de données à l'échelle du Web, c.à.d. répondre aux besoins croissants de centaines de millions d'utilisateurs et maintenant de plus en plus à des milliards de dispositifs connectés, y compris mais non limité aux mobiles, Smartphones, internet TV, et beaucoup plus.

De nos jours, le NoSQL est le terme désigné pour classer les BDs qui ne se basent pas sur le modèle du SGBD relationnel classique de nature ACID, et qui est – le NoSQL- inventé spécialement pour gérer la vitesse et l'extension des données web comme les sites Google, Facebook, Yahoo, Twitter et autres.

Les SGBD NoSQL essaient de trouver des solutions aux problèmes de passage à l'échelle et de la disponibilité des données en dépit de l'atomicité et la cohérence impliqués par les SGBD relationnels avec leurs transactions ACID.

Au contraire du SGBD relationnel, les unités des données ne sont pas bien définies en termes de type, taille et d'autres contraintes. Il n'y a pas de notion de colonnes et de lignes ou les colonnes ont des relations, et il n y a pas des transactions ACID.

Imaginons un utilisateur qui veut acheter un livre du site Amazon et la transaction est atomique, ce qui signifie que d'autres utilisateurs doivent attendre jusqu'à la fin de la transaction pour pouvoir lancer leurs commandes pour des raisons d'inventaire ; ce qui est impossible vu le nombre des commandes passant instantanément.

Amazon utilise les données en cache et même des enregistrements débloqués ce qui influence directement la cohérence.

Il convient de noter que dans le cas des transactions bancaires, il est impossible de travailler avec des données en cache ou avec un état temporaire vu la criticité (la nature critique) de la situation. Simplement, le NoSQL n'est pas toujours une solution.

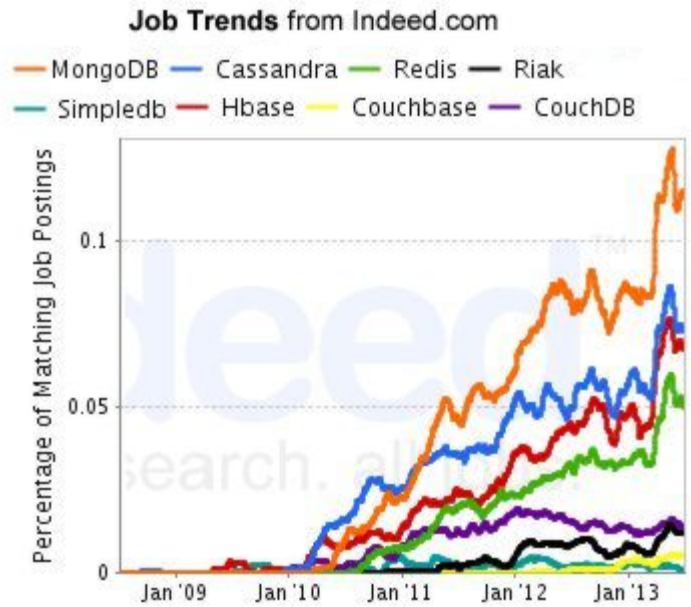

Figure 10- Les offres d'emploi NoSQL[5]

[5]Le site des offres d'emploi indeed.com

10. Caractéristiques

Le NoSQL n'est pas juste pour faire face au problème de l'extensibilité, mais il a proposé aussi d'autres solutions :

- Une représentation de données sans-schéma : Un nouveau technicien recruté au sein d'une compagnie doit connaitre d'abord tout le modèle d'une BD relationnelle existante à savoir les entités, leurs relations et le code pour pouvoir la maintenir. Par contre, avec une BD NoSQL, la représentation est plus flexible, nous ne sommes pas obligé à penser à la structure rigide de la BD, elle peut évoluer au cours de son développement, y compris l'ajout de nouveau champs et la fusion des données (JSON) et même une éventuelle intégration avec d'autre plateformes hétérogènes.
- Plus de requêtes complexes : Pas de requêtes SQL longues et des jointures complexes, ce qui réduit sensiblement le temps de développement et optimise l'utilisation des ressources.
- Vitesse : Gain de confiance des utilisateurs grâce à la vitesse de réponse (on parle de millisecondes au lieu de centaines de millisecondes)
- Mise à jour des données : Avec les SGBD relationnels, la mise à jour des données à travers des tables liées ou croisées est un scénario complexe, surtout si la table concernée ne fait pas partie de la transaction en cours, cela laisse la transaction ouverte pour une longue durée et ralentit la performance. Par contre, le NoSQL avec le principe de «Eventualy Consistant» permet des mises à jour concurrentes à travers plusieurs Datacenters avec la gestion des conflits et dans un temps très acceptable.
- Extensibilité (passage à l'échelle) planifiée : Nous n'avons pas à nous occuper de la taille de l'application ni du volume des données ni de leur schéma dès la conception. Par exemple, la prise en charge de dizaines de commandes en même temps sur un même produit par des clients distribués géographiquement ne peut être satisfaite par un SGBD classique.

11. Types

Les BD NoSQL sont classées selon le modèle de stockage de données:

Orientées documents, orientées clé-valeur, orientées colonnes, orientées graphe.

Les BDs Orientées colonnes

Au contraire des SGBDR, elles sérialisent les valeurs d'une colonne ensemble, puis les valeurs de la colonne suivante, etc. une BD relationnelle présente les données dans une table bidimensionnelle composée de lignes et colonnes, mais les manipule ligne par ligne, alors que le NoSQL orienté colonne stocke les données en tant que colonnes.

Dans une BDR, on stocke les données des fournisseurs comme suit :

	SupplierID	CompanyName	ContactName	ContactTitle	Address	City	Re
1	1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	49 Gilbert St.	London	N
2	2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	P.O. Box 78934	New Orleans	LA
3	3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	707 Oxford Rd.	Ann Arbor	M
4	4	Tokyo Traders	Yoshi Nagase	Marketing Manager	9-8 Sekimai Musashino-shi	Tokyo	N
5	5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra	Export Administrator	Calle del Rosal 4	Oviedo	A
6	6	Mayumi's	Mayumi Ohno	Marketing Representative	92 Setsuko Chuo-ku	Osaka	N
7	7	Pavlova, Ltd.	Ian Devling	Marketing Manager	74 Rose St. Moonie Ponds	Melbourne	Vi
8	8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	29 King's Way	Manchester	N
9	9	PB Knäckebröd AB	Lars Peterson	Sales Agent	Kaloadagatan 13	Göteborg	N
10	10	Refrescos Americanas LTDA	Carlos Diaz	Marketing Manager	Av. das Americanas 12.890	Sao Paulo	N

Figure 11- Table d'une BDR[6]

Alors que dans une BD NoSQL orientée colonne, elles sont stockées comme suit:

SupplierID	Company Name
1	Exotic Liquids
2	New Orleans Cajun Delights
3	Grandma Kelly's Homestead
4	Tokyo Traders

SupplierID	Contact Name
1	Charlotte Cooper
2	Shelley Burke
4	Yoshi Nagase

SupplierID	Contact Title
1	Purchasing Manager
2	Order Administrator
4	Marketing Manager

SupplierID	Address
1	49 Gilbert St.
3	707 Oxford Rd.
4	9-8 Sekimai Musashino-shi

SupplierID	City
1	New Orleans
2	
3	Ann Arbor
4	Tokyo

Figure 12 - Table d'une BD NoSQL orientée colonne

[6]Données de la Base AdventureWorks de Microsoft®

Il est conseillé d'entamer la modélisation en NoSQL par l'orientée colonne, car il permet une compréhension rapide du modèle et donne un avant-goût à ce nouveau domaine. Si le travail nécessite des valeurs agrégées, vous n'avez pas le choix à l'utiliser.

Avantages:

- Permet l'insertion facile de nouvelles colonnes à n'importe quel moment sans s'inquiéter des valeurs par défaut. Ceci assure une meilleure flexibilité au model et facilite le passage à l'échelle.
- La performance de ce modèle apparait clairement dans le calcul des valeurs maximale, minimale, moyenne et la somme.
- Si de nouvelles valeurs sont à appliquer sur l'ensemble des lignes ou sur un sous-ensemble de colonne, ce modèle permet un accès partiel aux données sans effet sur les données non concernées, ce qui accélère l'exécution.
- Optimisation de l'espace de stockage grâce aux types uniformes des colonnes, qui sont dans la plus part du temps des chaines de caractères de mêmes tailles. Tel caractéristique (ex : Chine comme pays pour 1 milliard d'utilisateurs) optimise la compression des données.

Les BDs orientées documents

Les enregistrements (les lignes du SGBDR) sont représentés par des documents ; ils sont semi-structurés par rapport à la représentation rigide du relationnel et permettent l'insertion, l'interrogation et la manipulation des données. Deux enregistrement peuvent avoir différentes structures ou ensemble de colonnes. Les enregistrements peuvent ne pas respecter un schéma spécifique ou une définition de table, ce que signifie qu'il n y a pas de validation de documents par rapport à un schéma comme c'est le cas pour les SGBDR.

En bref, une BD orientée document fournit une flexibilité dynamique ou un schéma modifiable ou complet des documents sans schéma.

Cet avantage a permis de rendre ce modèle plus répondu et plus utilisé parmi les autres modèles de BDs NoSQL.

Avec JSON, qui est l'un des langages qui adopte l'orienté-document, un document peut s'écrire comme suit :

```
{
     "SupplierID" : "1",
     "CompanyName" : "Exotic Liquids",
```

```
     "ContactName" : "Charlotte Cooper",
}
```

Un autre document dans la même base peut s'écrire comme suit :

```
{
     "SupplierID" : "2",
     "CompanyName" : "New Orleans Cajun Delights",
     "ContactName" : "Shelley Burke",
     "Adress":{
               "Line1"  :" P.O. Box 78934",
               "City" : "New Orleans",
               "Region" : "LA"
               },
     "Phones":["(100) 555-4822" , "(100) 555-4888"]
}
```

Un troisième document contient les informations d'un produit:

```
{
     "ProductID" : "9",
     "ProductName" : "Mishi Kobe Niku",
     "QuantityPerUnit" : "18 - 500 g pkgs.",
     "UnitPrice" : "97,00",
     "UnitsInStock" : "29",
     "UnitsOnOrder" : "0"
}
```

Nous constatons que les deux premiers documents sont presque similaires ; le deuxième est plus détaillé avec l'ajout de l'adresse du fournisseur. Alors que le troisième n'est pas en corrélation avec eux, ce sont des informations d'un produit.

Si nous devons travailler avec des agrégations à travers plusieurs entités, ce modèle nous permet un contrôle efficace sur la manière d'interroger les données. Exemple, travailler avec JSON à travers des données fusionnées ou avec XQuery en utilisant XML et obtenir des vues personnalisées.

Les BDs les plus connues de ce modèle sont : MongoDB, CouchDB, Jackrabbit, Lotus Notes, Terrastore, Redis et BaseX.

Avantages:

- Le contenu est flexible sans schéma
- Une recherche à travers multiple entités est négligeable par rapport à une même recherche dans un SGBDR classique.

Les BDs orientées clé-valeur

Ce modèle est proche du modèle orienté document, mais la création de la clé est obligatoire lors de la création du couple, il permet le stockage des données dans des couples clé-valeur. Absence de schéma ou de typage. La définition de la clé est obligatoire tandis que la valeur est opaque, pour cela, une clé doit être connue pour trouver la valeur associée. Son avantage est l'efficacité de travailler dans une mémoire distribuée ou dans un cache pour réduire les opérations d'E/S.

L'accès à une valeur est direct et efficace, car une paire clé-valeur est unique ; la complexité de son algorithme de recherche est de O(1). Les clés peuvent être indexées pour plus de performance.

La différence par rapport à une BD orientée document est l'absence de requêtes par rapport aux valeurs. L'interrogation des données se fait exclusivement par rapport aux clés.

Les pairs peuvent être imaginés comme une table à deux entrées :

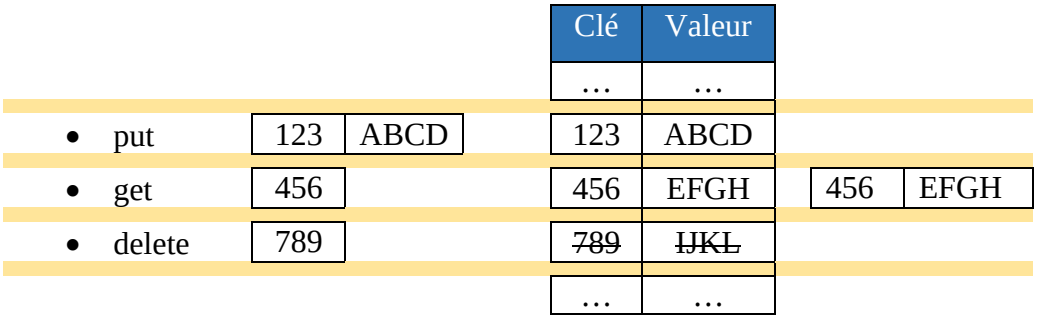

Figure 13 - Les commandes d'une BD clé-valeur

La plupart des BD orientées clé-valeur sont inspirées de Dynamo d'Amazon, qui garantit une évolutivité et une disponibilité exceptionnelles. Voldemort et Riak sont l'implémentation des bases de Dynamo.

Avantages:

- La recherche est optimale en raison de l'utilisation des clés et du cache.

Par exemple, Redis en fonctionnant sur un micro-ordinateur de gamme ordinaire peut scanner jusqu'à 1 million de clés dans moins de 40 milliseconde[7].

[7] http://redis.io/commands/keys

- Le type de données des valeurs n'est pas spécifié, on peut stocker n'importe quel type de données.

Ce modèle n'est pas conçu pour les applications nécessitant l'indexation des valeurs.

Les BDs orientées graphe

Ce modèle est basé sur la théorie des graphes, c.à.d. les nœuds, les relations et les propriétés. Il est relativement nouveau sur le marché des NoSQL.

Sa particularité est la facilité de définir les relations directement au niveau de la BD au contraire des autres modèles ou les relations sont visibles au niveau de l'application.

Il est très utile pour toutes les applications qui ont des relations complexes entre leurs objets comme les réseaux sociaux.

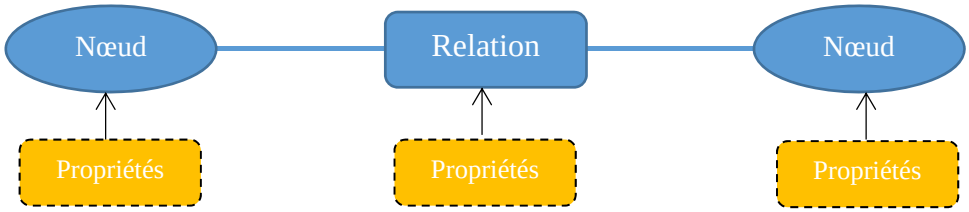

Figure 14- Structure de base d'une BD orientée graphe

Il profite de tous les avantages de la théorie des graphes :

- Trouver le chemin le plus court
- Trouver les voisins d'un nœud qui ont des propriétés spécifiques
- Quelle est la ressemblance entre deux nœuds en prenant en compte leurs voisins, etc…

Le W3C[8] utilise ce modèle dans son langage de représentation du web sémantique, le RDF[9].

Avantages:

- Ce modèle est idéal lorsque nous avons plusieurs objets liés les uns aux autres de façon complexe, et ses objets ont des propriétés (frère de, sœur de, père de, …). Il permet par une requête simple d'avoir les voisins d'un nœud, ou d'avoir tout un chemin par des requêtes plus ou moins complexes.

[8] The World Wide Web Consortium
[9] Resource Description Framework

- Il ne s'arrête pas sur le point de nous donner les relations entre les nœuds, mais aussi des rapports détaillés sur la nature de ses relations.
- Comme toute modélisation basée sur une représentation graphique, l'avantage majeur est la compréhension facile par les humains par rapport à une modélisation textuelle, c'est une représentation du monde réel, des noms, des villes, workstations (postes-de-travail) d'un réseau informatique, … ; l'insertion et la suppression des relations entre nœuds se fait par un simple clic de souris.

12. Liste des BDs NoSQL :

Ci-après la liste des fameuses BDs NoSQL les plus maturées et les plus puissantes sur le marché selon leurs modèles de données:

- Orientée Document: MongoDB, CouchDB, RavenDB et Terrastore
- Clé-valeur: Redis, Membase, Voldemort et MemcacheDB
- Orienté colonnes: Cassandra, BigTable, SimpleDB et Cloudera
- En graphe: Neo4J, FlockDB et InfiniteGraph

La plus part de ces BDs sont open source et Community Driven[10].

Les compagnies intéressées peuvent commencer avec un SGBD relationnel puis migrer vers un NoSQL; ou mieux, commencer directement avec un NoSQL, et le mieux serait de de mélanger le SGBBD relationnel avec le NoSQL.

« *L'exemple à citer, c'est Netflix, la compagne a migré d'Oracle vers Cassandra, elle atteint le million écriture par seconde en maintenant la moyenne de temps de réponse au 0.015 milliseconde, avec un coût total d'installation et de configuration sur le Cloud Amazon EC2 d'environ 60$ par heure pour un cluster de 48 nœuds et une capacité de stockage de 12.8 Téra-bytes, la bande passante est de 22 mbps en lecture et 18.6 mbps en écriture.* » (12)

13. Les BDs NoSQL hybrides:

Il existe d'autres BDs NoSQL qui supportent différents types de stockage, tels que :

- OrientDB: orientée document, orientée clé-valeur et orientée graphe.

[10] Développé par un groupe de développeurs indépendants.

- ArangoDB: une BD universelle qui peut être à la fois orientée document, orientée clé-valeur ou orientée graphe.
- Aerospike: une BD hybride entre le SGBDR et le NoSQL, elle est orientée document, orientée clé-valeur, orientée graphe et aussi un SGBDR.

Il est à noter que CouchDB et Neo4j et malgré qu'elles soient des NoSQL, elles respectent les transactions ACID.

14. Utiliser SGBDR ou NoSQL?

Le choix d'utiliser un SGBDR classique ou une BD NoSQL dépend fortement de la nature de l'application à développer.

14.1. Applications transactionnelles :

Les données : Elles ont une nature purement relationnelle. La performance de l'application est liée à la cohérence et l'intégrité des données. La concurrence est faible due à l'utilisation d'une seule BD et une seule instance à la fois.

La structure : bien définie à priori ; les types, les propriétés et les contraintes, les relations, les indexes … Le schéma n'est pas censé trop évoluer ou changer à l'avenir.

Accès aux données : cohérent, chaque lecture donne absolument la dernière mise à jour. L'accès aux données dans des tables croisées est fréquent.

NoSQL: Pour ce type d'applications, l'orientée colonne peut être utile dans la définition d'une structure évolutive dans le temps. L'orientée document lui aussi peut être utile dans l'implémentation des jointures ou la création des vues.

Néanmoins, il n y a pas de définition des relations, ni de la notion de transaction, a part le Neo4j, ni des propriétés ACID dans les transactions. Le support des jointures et les vues via l'orientée document pèsent sur la complexité des requêtes.

Décision: il est clair que pour telles applications, le choix de SGBDR est plus logique.

14.2. Applications de calcul

Ce sont les applications de statistiques, de sondage, …etc., où on a besoin de sous-ensembles de données et d'attributs. La plus part des opérations ne concernent que ce sous-ensembles d'attributs.

La structure : bien définie à priori ; les types, les propriétés et les contraintes, les relations, les indexes … le schéma n'est pas censé trop évoluer ou changer à l'avenir.

Accès aux données : L'Accès aux données est partiel, disant vertical d'une vision d'ensembles. Les données doivent être cohérentes tandis que quelque latence peut être permise et l'accès aux données dans des tables croisées est fréquent.

NoSQL : L'orientée colonne peut aider à définir une structure rigoureuse, aussi l'orientée document peut être utilisée ; les deux peuvent fournir de la vitesse et de l'évolutivité en traitant des données partielles. L'orientée document peut servir dans les jointures et la création des vues.

Par contre, à part l'orientée graphe, la création des relations peut être un casse-tête ; elles peuvent être crées et maintenues au niveau de l'application mais les données ne seront pas cohérentes.

Décision: Le cahier des charges peut être satisfait par un SGBDR ou par une BD NoSQL; mais les facteurs de la vitesse et de l'évolutivité donnent de grands avantages au NoSQL où les données peuvent être partitionnées horizontalement et verticalement.

14.3. Applications web

C'est le type le répondu de nos jours, il intéresse aussi bien les fournisseurs que les consommateurs en exploitant l'explosion du marché des gadgets et applications mobiles.

De telles applications doivent être obligatoirement capables d'évoluer et contenir de plus en plus de données pour absorber la demande croissante des utilisateurs et leur diversification géographique où un seul datacenter est insuffisant.

Les utilisateurs de ce type d'applications peuvent sacrifier un peu de cohérence de données pour satisfaire au temps de réponse.

Structure: Le schéma évolutif est une intégration avec d'autres applications est toujours envisagée. En aucun cas, les données existantes ne doivent être affectées. Les relations peuvent être optionnelles dans la couche BD ou dans la couche application.

Accès aux données : L'accès aux données est généralement partiel, les opérations CRUD doivent être faites dans de brefs délais et l'incohérence des données est tolérée pour une courte durée.

NoSQL: L'orientée document est un choix idéal pour un schéma flexible et évolutif, il peut fournir l'évolutivité voulue parce qu'il n'implémente pas les transactions ACID.

Décision: indifféremment, à utiliser le NoSQL.

15. L'utilisation du NoSQL

Les domaines les plus envahis par les BDs NoSQL sont:

- SaaS applications (CRM-ERP) flexible schéma: l'orientée document
- E-learning : L'orientée colonne
- Applications sociales (possibilité d'intégration) l'orientée document avec l'orientée colonne
- Relations (Twitter): L'orientée graphe

16. Travaux voisins

Vu la nouveauté du sujet, plusieurs études ont été menées sur les bases de données NoSQL durant les cinq dernières années.

En 2015, un article (1) a étudié le remplacement d'un SGBD relationnel par un NoSQL, en occurrence MySQL par MongoDB, et en conclusion, il a mis en cause l'extensibilité du MySQL.

En 2016, une thèse de projet de fin d'études en Master (2) a proposé une Etude critique des contextes ACID et BASE des bases de données NoSQL ; la conclusion du travail était une comparaison entre quatre bases sur les quatre critères ACID.

Dans la même année, un autre article(3) a mené une étude similaire mais en ajoutant d'autres critères tels que l'évolutivité, la performance théorique, la fiabilité, la flexibilité et la complexité.

Toujours en 2016 (3), une autre étude sur le NoSQL a envisagé plus de critères : à savoir ; la possibilité d'interactions, la vitesse, le type de données à stocker, le support à JSON, le recouvrement et la sécurité des données.

En 2018, une étude théorique comparative a été menée sur 10 systèmes Big Data (4) et leurs critères techniques servant à aider les professionnels à prendre la décision par rapport au choix à faire.

Plusieurs autres études et articles sont apparus traitant du même sujet ; répartis entre étude comparative SQL NoSQL, ou bien une étude comparative inter-NoSQL.

Notre étude sera différente car elle permettra la mise sur un banc d'essai pratique, le YCSB, en vue de tester des données réelles et avoir des résultats de comparaison entre deux fameuses bases de données NoSQL, MongoDB et Cassandra dans des configurations différentes et mesurer leurs performances respectives.

17. Conclusion

Apres avoir découvert les limites des SGBRs dans le premier chapitre, nous avons vu dans ce chapitre l'alternative qui a dominé le domaine de gestion de bases de données, et surtout dans les environnements distribués ; le NoSQL, qui a prouvé ses capacités dans la manipulation des grands volumes de données (documents, sites Web à fort trafic, etc.) à travers les modèles de données qu'il a proposé et les architectures qui assurent les critères demandés : la disponibilité et l'évolutivité. Nous avons constaté que la majorité des grandes entreprises du secteur informatique ont adopté ces nouvelles solutions pour répondre à leurs besoins majeurs. En fin de chapitre, nous avons cité quelques travaux voisins récents qui ont examiné le même sujet.

Chapitre III : Cassandra®, MongoDB® et YCSB

1. Introduction

Nous nous intéresserons particulièrement dans ce chapitre aux deux technologies du NoSQL: Cassandra et MongoDB. La raison pour laquelle nous avons choisis ces bases de données c'est que Cassandra est le type le plus populaire de base de données orienté colonne disponible sur le marché. Cassandra est largement utilisé, notamment par Facebook, c'est un programme open source et peut être exécuté sur plusieurs plates-formes, ce qui fait de lui une base de données orientée colonne de choix. De même, MongoDB est une base de données orientée document très utilisée. Elle implémente les fonctionnalités du document. Elle est aussi open source.

Enfin, nous présenterons le YCSB, l'outil qu'on va tester leurs performances avec, et nous parlerons de ces workloads qu'ils doivent être injecté dans ces bases de données pour mesurer leurs rondements.

2. Cassandra

2.1. Présentation

« Apache Cassandra est une base de données NoSQL puissante et considérablement extensible. Elle est désignée pour traiter des charges de BigData en temps réel à travers plusieurs Datacenters sans aucune défaillance. » (14)

Le nom Cassandra est inspiré d'une reine de la légendaire Troy qui pensait qu'elle fut un dépôt des évènements métaphysiques. Elle a été développée initialement par Facebook pour accélérer les recherches sur site. Avant d'être adoptée et distribuée comme open source par Google en 2008

Cassandra est une base de données orientée colonne, sa nature hautement disponible et hautement distribuée ne laisse aucun point de défaillance, cela veut dire que chaque nœud de son cluster peut répondre à n'importe quelle requête. Elle supporte les réplications sur plusieurs Datacenters.

Des géants du web utilisent Cassandra, tel que Netflix, eBay, Twitter, Reddit et Ooyala. « *À ce jour, le plus grand cluster publique Cassandra a plus de 300 Téra de données distribués sur 400 machines* » (14)

2.2. Le Modèle de données

Dans Cassandra, le Keyspace est le conteneur des données de l'application (un peu comme une Database ou un schéma pour une base de données relationnelle). Dans ces keyspaces se trouvent une ou plusieurs familles de colonnes (qui correspondent aux tables en base de données relationnelle).

Ces familles de colonnes contiennent des colonnes ainsi qu'un ensemble de colonnes connexes qui sont identifiées par une clé de ligne. En outre, chaque ligne d'une famille de colonnes ne dispose pas nécessairement des mêmes colonnes qu'une autre ligne.

Pour toutes les familles de colonnes, chaque ligne est identifiée de manière unique par sa clé (un peu comme la notion de clé primaire pour les bases de données relationnelles). Une famille de colonnes est, quant à elle, toujours partitionnée par ses clés de ligne qui sont, implicitement, indexées.

En fait, dans Cassandra, une colonne est le plus petit incrément de données. Elle est modélisée par un tuple qui contient le nom, la valeur et un timestamp. Ce timestamp est utilisé par Cassandra pour déterminer la mise à jour la plus récente dans la colonne (et, dans ce cas, c'est la plus récente qui est prioritaire lors d'une requête).

Les requêtes dans Cassandra sont statiques, nous devons connaitre leurs modèles avant la création du modèle de données ; ce modèle de données n'a pas les mêmes objectifs du modèle relationnel, à savoir le stockage efficace et les relations entre les entités, mais son objectif sous Cassandra est d'optimiser la performance des lignes et le stockage de grands volumes de données.

2.3. Caractéristiques

- **Sans-schéma :** Comme toutes les autres NoSQL, avec Cassandra vous pouvez directement commencer par l'insertion des données sans le casse-tête de la désignation de la structure. Après, chaque colonne est indépendante et peut avoir des lignes déférentes des autres ; c'est la flexibilité totale. Si vous avez a priori une idée claire sur la structure et les types de données à implémenter, Cassandra vous permet de structurer vos données comme la méthode relationnelle classique.
- **CQL :** «Cassandra Query Language», est un langage proche du traditionnel SQL mais sans les fonctions GROUP BY, JOIN et avec une utilisation limitée de

ORDER BY ; elles sont remplacées par d'autres plus améliorées spécialement pour Cassandra.

Exemple:

```
cqlsh> CREATE KEYSPACE test with strategy_class =
       'SimpleStrategy' and
       Strategy_options:replication_factor=1;

cqlsh> USE test;

cqlsh> CREATE COLUMNFAMILY users (
       key varchar PRIMARY KEY,
       full_name varchar,
       birth_date int,
       state varchar );

cqlsh> CREATE INDEX ON users (birth_date);

cqlsh> CREATE INDEX ON users (state);

cqlsh> INSERT INTO users (key, full_name, birth_date,
       state) VALUES ('bsanderson', 'Brandon Sanderson',
       1975,'UT');

cqlsh> INSERT INTO users (key, full_name, birth_date,
       state) VALUES ('prothfuss', 'Patrick Rothfuss',
       1973, 'WI');

cqlsh> INSERT INTO users (key, full_name, birth_date,
       state) VALUES ('htayler', 'Howard Tayler', 1968,
       'UT');

cqlsh> SELECT key, state FROM users;
     key | state |
     bsanderson | UT |
     prothfuss | WI |
     htayler | UT |

cqlsh> SELECT * FROM users WHERE state='UT' AND birth_date
     > 1970;
     KEY | birth_date | full_name | state |
     bsanderson | 1975 | Brandon Sanderson | UT |
```

- **Cluster :** Deux ou plus d'instances (nœuds) Cassandra travaillant ensemble ; elles communiquent en utilisant un protocole appelé Gossip.
- **Environnement homogène :** Tous les nœuds de Cassandra sont semblables et contiennent chacun toute la configuration nécessaire pour compléter le cluster ; à

l'encontre des autres bases de données NoSQL comme HBase qui a des nœuds hétérogènes.

- **Quorum :** C'est le nombre de nœuds qui répond lors d'une requête ; il est basé sur le RF ; un quorum lecture/écriture signifie que la requête interroge RF/2+1 nœuds ; pour RF = 3, une requête interroge 2 nœuds. Ce qui assure la cohérence des données.
- **Facteur de réplication (RF) :** C'est le nombre de copie de chaque ligne de la base qu'on décide de sauvegarder dans chaque cluster; RF de 3 signifie que 3 copies de la base résident dans le cluster.

  Il est aussi utilisé pour déterminer le nombre de quorums, ce qui assure la disponibilité et la tolérance aux défaillances. Dans Cassandra, il n y a pas de copie maîtresse et copie esclave, toutes les copies ont la même valeur vis-à-vis du cluster. Une règle à respecter lors de la création des copies, est que le nombre de copies ne doit pas dépasser le nombre de nœuds disponibles dans le cluster, sinon, on risque de ne pas pouvoir écrire sur la base de données, seules les lectures restent possibles pour des raisons de disponibilité.
- **Cohérence réglable :** Cassandra permet le réglage du niveau de cohérence selon le besoin en lecture/écriture. Le plus faible est le niveau «ANY» tandis que «ALL» est la plus forte cohérence. Ces niveaux sont:
  - ANY: le plus faible, write ou update doit réussir dans n'importe quel nœud\replica disponible. Une écriture doit être écrite sur au moins un nœud. Si tous les nœuds de réplique de la clé de ligne donnée sont en panne, l'écriture peut toujours réussir une fois qu'un transfert a été écrit.
  - ONE: write\update doit réussir dans au moins un nœud responsable de cette ligne (primaire ou réplique).
  - QUORUM: write\update affecte la majorité des répliques, c'est-à-dire doit réussir dans un nombre minimum de nœuds réplicas déterminé par RF/2+1.

  D'autres réglages sont permis.

- **Partitionner :** C'est le mécanisme qui détermine la stratégie de l'emplacement des réplications ; c'est une simple fonction de hachage. Il calcule le numéro de jeton à assigner pour chaque ligne qu'il utilise pour trouver cette ligne

ultérieurement. Différents algorithmes sont utilisés : tri par ordre alphabétique, tri au hasard …etc.

- **Stratégies de réplication :**
  - **Stratégie simple:** utilisée avec une seule machine, par conséquent, elle est l'emplacement par défaut de l'espace de stockage. Le Partitionner est le responsable de la distribution des réplications sur les nœuds. La première réplication est disposée d'une manière aléatoire, puis les autres sont ajoutées dans le sens des aiguilles d'une montre. Tous les nœuds sont sur la machine ou le datacenter local ; et une fois lancée la requête n'a pas à chercher ailleurs.
  - **Stratégie de la topologie de réseau:** utilisée lors du déploiement de Cassandra sur plusieurs machines ou Datacenters d'un cluster. Elle détermine le nombre de réplications à créer sur chaque Datacenter.
- **Le nombre de réplications :** Lorsqu'on décide du nombre de réplications à créer dans chaque nœud, il y a deux critères à prendre en considération :
  - **La latence :** Si une lecture doit être faite sur un autre Datacenter que celui sollicité.
  - **Les défaillances acceptables :** Les scenarios à prévoir si un nœud tombe en panne

  On peut avoir un cas où le nombre de réplications diffère d'un Datacenter à un autre ; lorsque le nombre de lectures sur un Datacenter est très élevé par rapport à un notre, il est conseillé d'augmenter le nombre de réplications sur le premier pour réduire la latence.

- **Snitches :** Un snitch est le protocole qui veille à déterminer quel nœud doit être sollicité lors d'une lecture par la création d'une topologie qui rassemble les nœuds ensemble.
- **Déploiement :** Cassandra peut être exécutée sur tout système qui supporte la machine virtuelle de Java JVM, mais pour profiter au maximum de sa performance, il est vivement conseillé d'utiliser un service Cloud (Amazon Web Service, Google Compute Engine, Windows Azure…).

3. MongoDB

3.1. Présentation

Le nom de MongoDB est dérivé de l'adjectif anglais «hu**mongo**us» **D**ata **B**ase, qui signifie énorme ou gigantesque base de données. C'est l'une des bases de données NoSQL, ou il n y a pas les concepts de table, schéma, requête SQL, jointure, clé étrangère et les propriétés ACID. MongoDB est une base de données orientée document, son slogan est «*une taille ne convient pas tous*» (13), elle utilise les documents au lieu des lignes classiques, et c'est grâce à ce changement qu'on verra qu'elle est plus rapide, considérablement plus évolutive horizontalement et plus simple à utiliser. Pour aboutir à ce résultat, MongoDB a sacrifiée de quelques propriétés, par conséquent, elle n'est pas idéale pour toutes les situations.

La notion de cohérence est relative, c'est pourquoi MongoDB n'est pas une solution pour les applications de gestion bancaire.

L'un des principaux concepts de La MongoDB est quelle est basée sur plus d'une copie sur différents serveurs, c'est l'une tombe en panne, il y aura toujours une copie de restauration. Elle est adaptée pour tous les systèmes d'exploitation, Windows, Linux, Solaris et Mac OS.

L'utilisateur n'a pas à se soucier de la façon d'organiser ses données, il suffit de les mettre ensemble dans un document et les stocker dans MongoDB.

MongoDB utilise le BSON format pour stocker les données, BSON désigne l'abréviation de «Binary JSON» ; JSON «Java Script Object Notation» permet la représentation de données complexes et structurées d'une manière simple et lisible par l'être humain, il est à la fois un langage d'échange et de stockage de données. Parce qu'il stock toutes les données liées dans un seul document et tous les documents similaires dans une seule place, BSON permet de rendre le traitement des requêtes plus rapide.

3.2. Le sharding

Le sharding, la fragmentation, est une technique qui permet l'extension horizontale d'un système sur multiples serveurs (stockage distribué). Il nous permet de dépasser les limitations matérielles par la propagation des données sur plusieurs partitions physiques (shards).

Les documents de MongoDB peuvent être répartis sur plusieurs de serveurs, chaque serveur vérifie s'il contient l'information voulue et retourne le résultat. Grâce au concept de sharding, MongoDB traite une requête sur deux serveurs avec la même facilité et efficacité sur une centaine de serveurs. Il prend en charge les opérations de fractionnement et recombinaison des données en maintenant la transparence du côté des développeurs. Ce concept donne des ailes aux possibilités de scalabilité de MongoDB.

3.3. Caractéristiques

- Le MongoDB est développé avec le C++, c'est pourquoi il est exécutable partout sur toutes les plateformes, que ce soit 32 ou 64 bits et avec des exigences minimales.
- Il est exigé que chaque document soit identifié de façon unique, si l'identifiant n'est pas créé par l'utilisateur, MongoDB le génère automatiquement. Tous les documents sont indexés sur le champ _id ; ce champ ne peut pas être supprimé.
- Le contenu d'un document est un ensemble de paires clé-valeur ; au contraire des SGBBRs, quand la valeur est inconnue, simplement, la clé est omise.

```
{ "Nom" : "Benali" , "Prénom" : "Ahmed" ,
"Telephones" : ["032 32 32 32" , "062 26 62 26"]}
```

- **Les collections:** MongoDB permet la création des collections d'articles similaires ou d'articles mixtes selon le besoin ; pour des raisons d'indexation, il est préférable d'utiliser des collections d'articles identiques.
- MongoDB est une collection de collections de documents hétérogènes, ce qui lui donne la souplesse et la facilité de créer les bases de données.
- **Des requêtes dynamiques :** bizarre comme critère, parce que d'autres BD NoSQL ne supportent pas des requêtes dynamiques telle que CouchDB.
- **L'indexation géospaciale :** introduite récemment, c'est la possibilité d'indexer des items selon leurs distances de coordonnées d'un point géographique donné.
- **Optimisateur de requêtes :** MongoDB dispose d'un outil interne qui peut aider à optimiser les requêtes clients en optimisant le temps de réponse.
- La mise à jour des données se fait directement dans la mémoire, ce qui signifie que MongoDB n'a pas besoin d'extra mémoire ni d'écriture sur disque ; ce qui lui offre plus de vitesse d'exécution mais moins de fiabilité de données.
- Replica sets: Ensemble de copies identiques de bases de données ou une seule est master (maîtresse/maître) et les autres sont (slaves) esclaves, l'écriture ne se fait

que sur la copie maîtresse et ensuite une mise à jour est opérée sur les copies esclaves. Si une défaillance survient sur la copie maîtresse, l'une des copies esclaves est élue maîtresse ; ce qui garantit la durabilité de la base.

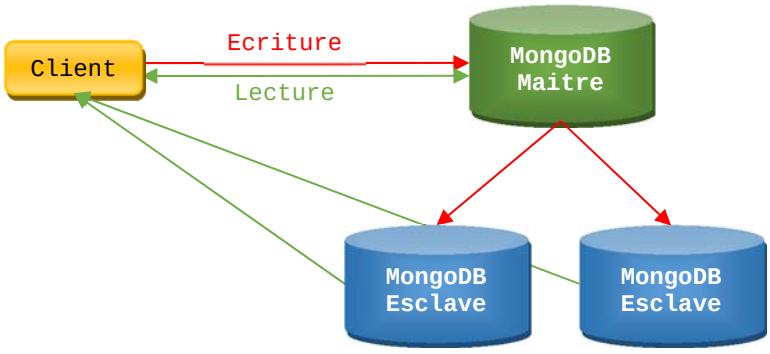

Figure 15 - Le concept de Replica set de MongoDB

4. YCSB

YCSB (Yahoo! Cloud Serving Benchmark (15)) est un banc d'essai open source qui a pour objectif l'évaluation des performances des bases de données NoSQL et les Data Warehouse Clouds. Initialement Yahoo! L'ont développé en 2010 pour comparer leur propre SGBD NoSQL PENUTS avec les autres SGBD NoSQL.

Il s'occupe de deux volets : la performance et l'extensibilité. La performance d'une base de données est un ensemble de critères mesurables et qui peuvent différencier une BD de l'autre. L'un de ces critères est la latence des opérations, ou bien s'est le temps de réponse des requêtes des utilisateurs ; le deuxième critère c'est le temps d'exécution d'un workload (ensemble d'Operations), et le troisième critère le nombre d'opération par seconde (le débit d'opération). La performance est le volet qui préoccupe plus les utilisateurs lors de la navigation sur le web. Dans ce volet, le YCSB, à travers son générateur de charge de travail (The workload generator), génère un dataset et le charge dans les bases de données à tester, puis il exécute les opérations de mesure de la performance. Les résultats obtenus sont essentiellement le nombre et le taux d'opérations CRUD exécuté avec succès (lecture, écriture, mise à jour ou insertion) et la latence moyenne de chaque opération. Ces résultats peuvent être représentés sous forme de graphe pour mieux comprendre, analyser et comparer.

Les tests d'extensibilité se font par l'exécution d'un même workload sur un SGBD puis comparer ses mêmes résultats en ajoutant des serveurs au système. Si nous déployons des nouvelles instances (serveurs) et le système en arrêt, nous parlons de scalabilité (extensibilité) ; si nous déployons des instances et le système en marche, nous parlons d'élasticité.

Nous l'utiliserons pour tester des différents scénarios de charges de travail (workloads) sur un même jeu de données dans les deux BD NoSQL, Cassandra, et MongoDB.

4.1. Workloads

Les charges de travail (workloads) sont des variations des opérations de lecture, d'écriture, d'insertion et de mise à jour des datasets. Les données créées sont des données aléatoires ; tandis que la taille d'un enregistrement, le nombre d'enregistrements et le nombre d'opérations sont paramétrables.

Les opérations possibles par YCSB sont :

- Insert : Inserer un nouveau enregistrement
- Update : mettre à jour un enregistrement
- Read : Lire un enregistrement
- Scan : Scanner un sous ensemble d'enregistrements, l'enregistrement de début et le nombre d'enregistrement à scanner sont choisis au hasard.

En plus, le choix de données à lire ou à mettre à jour est lié à la méthode de sélection d'enregistrement cible ; ces méthodes sont :

- Uniforme : Tous les enregistrements ont le même poids lors du choix
- Distribution de Zipfian : Classer les enregistrements selon leur popularité (apparition)
- Le dernier : Classer les enregistrements par leur date d'insertion, les derniers insérés sont les premiers dans la distribution.

Les workloads prédéfinis par YSCB (14) :

Workload	Opérations	Méthode
A- Mise à jour lourde	Lecture : 50% MàJ : 50%	Zipfian
B- Lecture lourde	Lecture : 95% MàJ : 5%	Zipfian
C- Lecture seule	Lecture : 100%	Zipfian
D- Lecture des derniers	Lecture : 95% Insertion : 5%	Le dernier
E- Petits plages	Scan : 95% Insertion : 5%	Zipfian / Uniforme
F- Lecture-MàJ-Ecriture	33% - 33% - 33%	Zipfian

Les workloads ne sont pas limité à ces 6 prédéfinis, d'autres sont disponibles, et l'utilisateur peut définir d'autres workloads appropriés à ces besoins.

Le benchmark fournit des couches d'interface à presque toutes les BD connues, Cassandra, MongoDB, HBase, Accumulo et Voldemort. Neanmoins l'utilisateur peut l'étendre à d'autres bases de donnees.

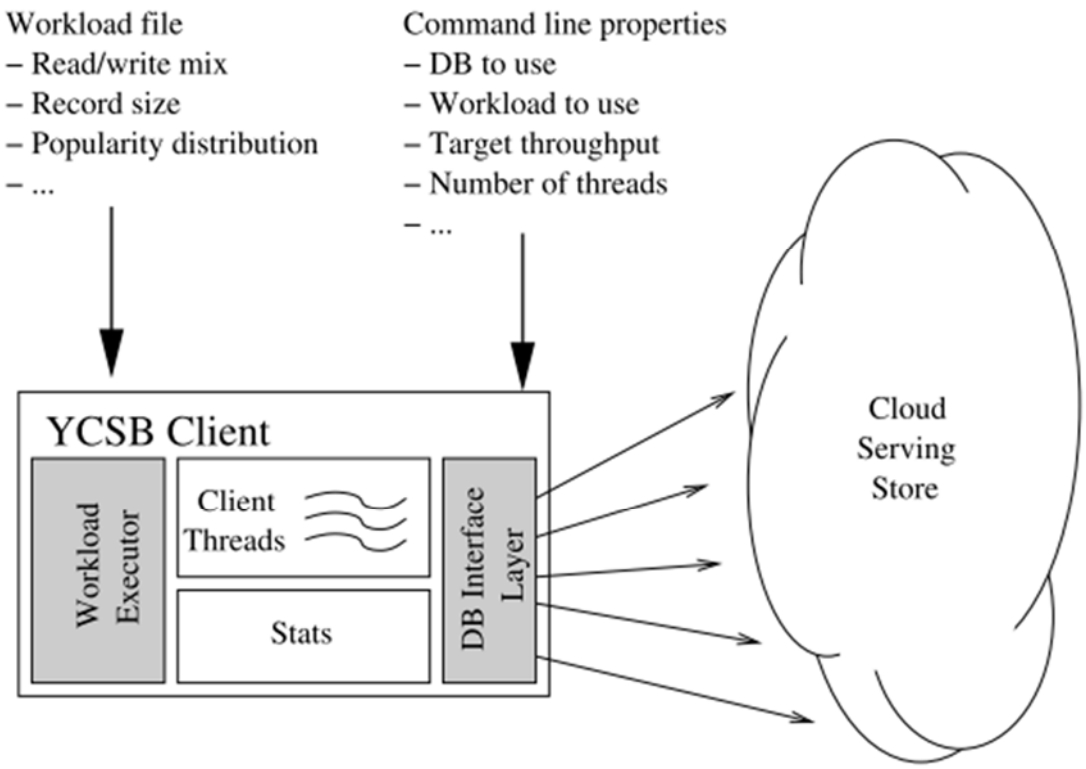

Figure 16 - Architecture de YCSB

4.2. Installation et exécution

L'installation de YCSB sous Windows est facile, il suffit de télécharger le répertoire sous une forme zip de sa source et le décompresser sur le disque dure (de préférence dans une racine c:\ ou d:\ pour faciliter l'exécution. Les etapes d'execution sont comme suit :

- Installer la base de données à tester
- Créer dans cette BD une BD « ycsb » (keyspace pour Cassandra), puis une table « usertable » (Collection pour MongoDB) dans cette BD.

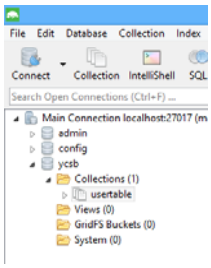

Figure 17 - Une BD ycsb et une table usertable pour l'execution de YCSB

- Choisissez le workload à tester
- Choisissez les paramètres d'exécution, tels que la méthode de distribution, le nombre de threads, le nombre d'enregistrements, le ombre de champs dans un enregistrement …

- Lancez la commande ycsb

```
C:\ycsb-0.15.0>bin\ycsb load cassandra-cql -p hosts=localhost
    -threads 10 -p operationcount=100000 -p
    recordcount=300000 -p fieldcount=10 -P
    workloads/workloadc -s >wlc_load_5nodes.txt
```

- Une fois terminée, les résultats sont affichés sur le terminal DOS ou a priori dans un fichier texte.

```
[OVERALL], RunTime(ms), 339319
[OVERALL], Throughput(ops/sec), 294.7079297062646
 [READ], Operations, 49998
[READ], AverageLatency(us), 24960.595723828952
[READ], MinLatency(us), 525
[READ], MaxLatency(us), 60293119
[READ], 95thPercentileLatency(us), 74687
[READ], 99thPercentileLatency(us), 481023
[READ], Return=OK, 49998
[UPDATE], Operations, 50002
[UPDATE], AverageLatency(us), 41474.25864965402
[UPDATE], MinLatency(us), 666
[UPDATE], MaxLatency(us), 60751871
[UPDATE], 95thPercentileLatency(us), 186879
[UPDATE], 99thPercentileLatency(us), 579071
[UPDATE], Return=OK, 50002
```

5. Conclusion

Dans ce chapitre nous avons décrit les architectures, les caractéristiques techniques et les composants des deux des célèbres systèmes NoSQL, Cassandra et MongoDB. Afin de donner une idée illustrative des performances de ces BD NoSQL, nos avons présenté le YCSB de Yahoo!, le banc d'essai que nous utiliserons dans le prochain chapitre.

IMPLEMENTATION ET EXPERIMENTATION

1. Introduction

Ce chapitre est consacré à la partie expérimentale de l'étude. Nous définirons le banc d'essai YCSB, son implémentation et son utilisation ; nous citerons l'environnement du test, les différents configurations et implémentations des outils utilisés, et nous analyserons les résultats issus de la pratique des différentes expériences.

2. L'expérience

Les tests de YCSB ont été faits sur une machine avec les caractéristiques suivantes :

- Processeur : Intel® Core™ i5-3210M CPU @ 2.50 GHz x 2
- RAM : 6 GB
- Système : Microsoft Windows8.1 Pro 64-bit
- Disque dur: 500 GB

3. Configuration de Cassandra :

Nous avons installé la dernière version de Cassandra à ce jour, à savoir la version 3.11.3; la configuration minimale requise est la création de 3 répertoires, data pour les données, log pour les journaux et saved_caches pour les caches. Les configurations hors la configuration par défaut se font dans le fichier bin\cassandra.yaml.

Des outils d'aide sont disponibles tel que l'OpsCenter, un outil visuel web-based qui simplifie les opérations sur les clusters, l'insertion et la suppression des nœuds, jointure des clusters, l'état du cluster ou des nœuds.

Figure 18 - OpsCenter de Cassandra

Un autre outil fournis, c'est le DevCenter, il est utile pour le traitement des données (les opérations CRUD).

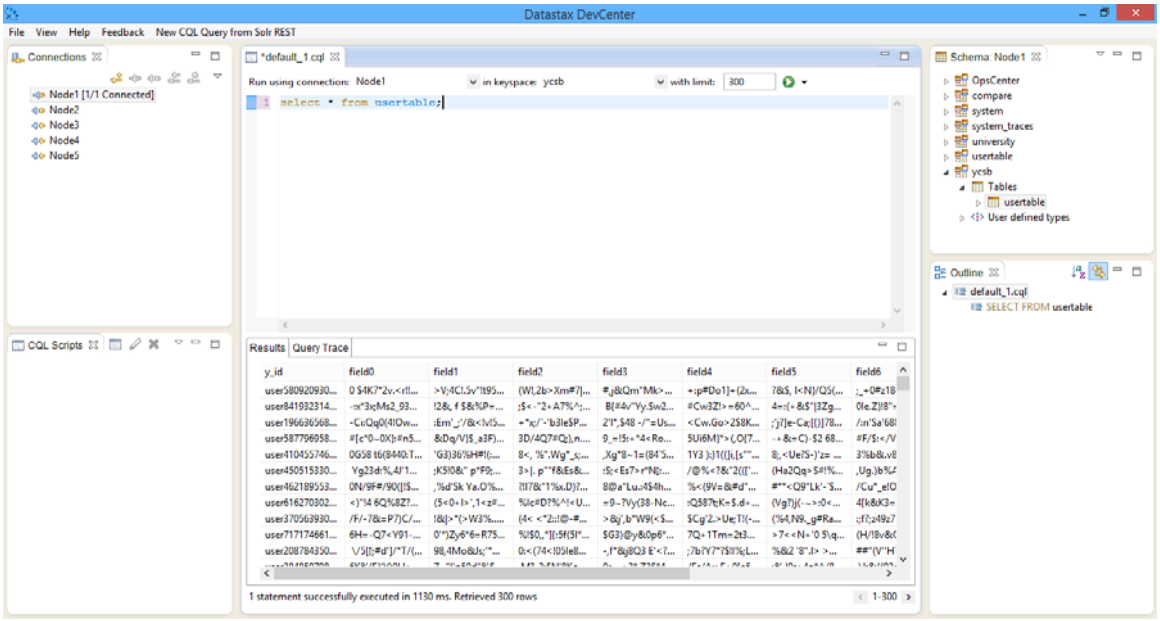

Figure 19 - DevCenter de Cassandra

Aussi, Cassandra Cluster Manager CCM, un excellent outil pour créer et gérer rapidement un cluster local sur une machine. Il dépend de l'installation de Python et de Ruby. Nous l'avons utilisé pour créer rapidement des nœuds locaux, exemple :

```
C:> ccm create TestCluster -v 2.1.2 -n 10
```

Cette ligne de commande crée un cluster local de Cassandra version 2.12 avec 10 nœuds.

4. Configuration de MongoDB

Pour MongDB aussi, nous avons installé la dernière version stable, 4.0.0. À l'instar de Cassandra, elle est menée d'une interface graphique web-based, l'OpsManager pour la gestion des clusters et replicasets et le shards et Studio 3T pour les opérations CRUD.

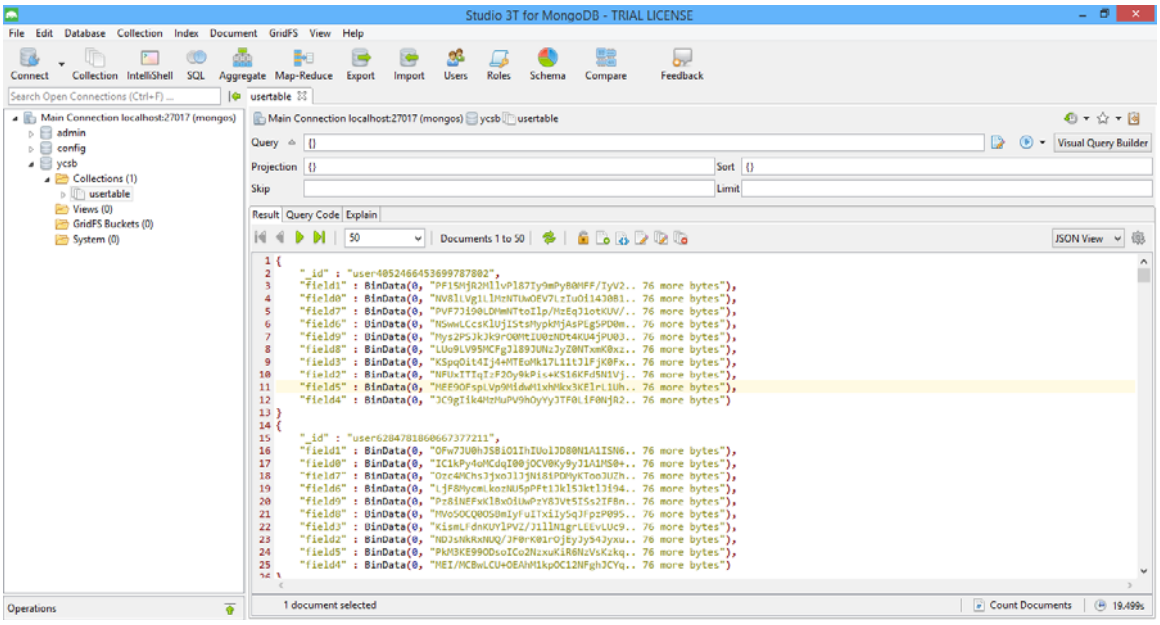

Figure 20 - Studio 3T de MongoDB

Aussi, MTools, une collection de scripts pour implémenter localement les environnements complexes. Nous l'avons utilisé pour implémenter les différents environnements : 1, 3, 5 et 10 shards avec 3 replicasets pour chaque shard. Exemple d'utilisation :

```
C :> mlaunch init --replicaset --sharded 3
```

Cette ligne de commande crée un cluster MongoDB avec 3 shards.

5. Configuration de YCSB

Les 6 workloads prédéfinis de YCSB ont été testés sur les deux SGBD Cassandra et MongoDB, avec 1, 3, 5 et 10 nœuds (shards pour MongoDB) ; chaque workload doit être chargé puis exécuté. La moyenne de temps de chaque opération été de 10 minutes, tous les essais ont duré presque 8 heures.

Notre test a été paramétré sur les SGBD pour tous les workloads comme suit :

10 utilisateurs (instances), 100000 opérations sur 300000 enregistrements de 10 champs. Ce choix est jugé par la moyenne de temps d'exécution de chaque workload.

Exemple d'exécution d'une commande YCSB:

```
C:\ycsb-0.15.0>bin\ycsb load cassandra-cql -p
hosts=localhost -threads 10 -p operationcount=100000 -p
recordcount=300000 -p fieldcount=10 -P workloads/workloadc
-s >wlc_load_5nodes.txt
```

- bin\ycsb : lance le benchmark

- cassandra-cql : la couche d'interface avec la base de données, pour MongoDB, c'est mongdb
- hosts : l'adresse IP du serveur, ici c'est le serveur local localhost
- operationcount : le nombre d'opérations à exécuter
- recordcount : le nombre d'enregistrements
- fieldcount : le nombre de champs dans un enregistrement
- workloads\workloadc : le workload à tester, ici c'est le workload C
- -s > wlc_load_5nodes.txt: enregistrer les résultats dans fichier.txt

Exemple de déroulement de l'exécution :

```
2018-09-11 16:17:28:889 1850 sec: 94310 operations; 0
current ops/sec; est completion in 1 minute [READ:
Count=0, Max=0, Min=9223372036854775807, Avg=?, 90=0,
99=0, 99.9=0, 99.99=0] [READ-MODIFY-WRITE: Count=0, Max=0,
Min=9223372036854775807, Avg=?, 90=0, 99=0, 99.9=0,
99.99=0] [UPDATE: Count=0, Max=0, Min=9223372036854775807,
Avg=?, 90=0, 99=0, 99.9=0, 99.99=0]
```

```
 Max=954367, Min=610, Avg=11742,33, 90=9735, 99=337407, 99.9=914943, 99.99=926207]
2018-09-10 14:19:32:661 60 sec: 66114 operations; 954,93 current ops/sec; est completion in 3 minutes [INSERT: Count=9562,
 Max=2400255, Min=605, Avg=11409,4, 90=8847, 99=232575, 99.9=1098751, 99.99=2365439]
2018-09-10 14:19:42:654 70 sec: 79022 operations; 1291,57 current ops/sec; est completion in 3 minutes [INSERT: Count=1290
2, Max=582655, Min=612, Avg=7464,92, 90=10079, 99=167295, 99.9=498175, 99.99=577535]
2018-09-10 14:19:52:653 80 sec: 89376 operations; 1035,5 current ops/sec; est completion in 3 minutes [INSERT: Count=10354
, Max=827903, Min=607, Avg=9980,31, 90=10055, 99=257023, 99.9=678399, 99.99=799231]
2018-09-10 14:20:02:653 90 sec: 96990 operations; 761,4 current ops/sec; est completion in 3 minutes [INSERT: Count=7614,
Max=939007, Min=598, Avg=13134,05, 90=9191, 99=475647, 99.9=889855, 99.99=920063]
2018-09-10 14:20:12:653 100 sec: 106972 operations; 998,2 current ops/sec; est completion in 3 minutes [INSERT: Count=9982
, Max=848383, Min=605, Avg=9480,42, 90=9671, 99=199807, 99.9=526847, 99.99=813567]
2018-09-10 14:20:22:653 110 sec: 115464 operations; 849,2 current ops/sec; est completion in 2 minutes [INSERT: Count=8493
, Max=1119231, Min=600, Avg=12403,34, 90=9543, 99=294143, 99.9=1073151, 99.99=1081343]
2018-09-10 14:20:32:653 120 sec: 121289 operations; 582,5 current ops/sec; est completion in 2 minutes [INSERT: Count=5826
, Max=1470463, Min=619, Avg=17151,3, 90=11831, 99=463103, 99.9=818687, 99.99=830463]
2018-09-10 14:20:42:653 130 sec: 126756 operations; 546,7 current ops/sec; est completion in 2 minutes [INSERT: Count=5465
, Max=2916351, Min=595, Avg=18260,64, 90=9295, 99=533503, 99.9=956927, 99.99=2910207]
2018-09-10 14:20:52:653 140 sec: 141788 operations; 1503,2 current ops/sec; est completion in 2 minutes [INSERT: Count=150
32, Max=488959, Min=600, Avg=6636,22, 90=10207, 99=136575, 99.9=299519, 99.99=450303]
2018-09-10 14:21:02:654 150 sec: 155678 operations; 1388,86 current ops/sec; est completion in 2 minutes [INSERT: Count=13
890, Max=521727, Min=620, Avg=7188,42, 90=10599, 99=141951, 99.9=422911, 99.99=483839]
2018-09-10 14:21:12:653 160 sec: 165615 operations; 993,8 current ops/sec; est completion in 2 minutes [INSERT: Count=9940
, Max=727551, Min=608, Avg=10094,68, 90=10487, 99=245887, 99.9=633855, 99.99=697855]
2018-09-10 14:21:22:653 170 sec: 177001 operations; 1138,6 current ops/sec; est completion in 1 minute [INSERT: Count=1138
3, Max=740351, Min=596, Avg=8618,43, 90=9855, 99=169215, 99.9=655871, 99.99=702975]
2018-09-10 14:21:32:654 180 sec: 188519 operations; 1151,68 current ops/sec; est completion in 1 minute [INSERT: Count=115
```

Figure 21 - Exemple d'un workload d'insertion

```
2018-09-10 14:54:42:297 360 sec: 40784 operations; 39,5 current ops/sec; est completion in 8 minutes [READ: Count=376, Max
=8675327, Min=550, Avg=265235,19, 90=265727, 99=8384511, 99.9=8675327, 99.99=8675327] [UPDATE: Count=19, Max=8351743, Min=
870, Avg=505251,63, 90=259455, 99=8351743, 99.9=8351743, 99.99=8351743]
2018-09-10 14:54:52:297 370 sec: 43812 operations; 302,8 current ops/sec; est completion in 7 minutes [READ: Count=2877, M
ax=406527, Min=523, Avg=33137,03, 90=127999, 99=232447, 99.9=289535, 99.99=406527] [UPDATE: Count=151, Max=233727, Min=858
, Avg=28098,91, 90=100799, 99=176127, 99.9=233727, 99.99=233727]
2018-09-10 14:55:02:298 380 sec: 47534 operations; 372,16 current ops/sec; est completion in 7 minutes [READ: Count=3527,
Max=429311, Min=521, Avg=26785,74, 90=98815, 99=255743, 99.9=349439, 99.99=429311] [UPDATE: Count=195, Max=327679, Min=825
, Avg=28329,13, 90=114815, 99=326143, 99.9=327679, 99.99=327679]
2018-09-10 14:55:12:297 390 sec: 51835 operations; 430,14 current ops/sec; est completion in 6 minutes [READ: Count=4080,
Max=350207, Min=515, Avg=22721,26, 90=88191, 99=242303, 99.9=327935, 99.99=350207] [UPDATE: Count=221, Max=240511, Min=825
, Avg=22705,29, 90=85119, 99=218879, 99.9=240511, 99.99=240511]
2018-09-10 14:55:22:298 400 sec: 54025 operations; 218,98 current ops/sec; est completion in 5 minutes [READ: Count=2080,
Max=1251327, Min=519, Avg=47366,11, 90=106303, 99=751103, 99.9=1145855, 99.99=1251327] [UPDATE: Count=110, Max=693759, Min
=853, Avg=38112,82, 90=83199, 99=632319, 99.9=693759, 99.99=693759]
2018-09-10 14:55:32:297 410 sec: 58230 operations; 420,54 current ops/sec; est completion in 4 minutes [READ: Count=3972,
Max=697855, Min=523, Avg=24235,87, 90=73983, 99=400127, 99.9=636415, 99.99=697855] [UPDATE: Count=233, Max=280831, Min=853
, Avg=17642,14, 90=63903, 99=225151, 99.9=280831, 99.99=280831]
2018-09-10 14:55:42:297 420 sec: 64459 operations; 622,9 current ops/sec; est completion in 3 minutes [READ: Count=5908, M
ax=407551, Min=518, Avg=15771,77, 90=63967, 99=182015, 99.9=396543, 99.99=405247] [UPDATE: Count=321, Max=401151, Min=857,
 Avg=19908,9, 90=64767, 99=265215, 99.9=401151, 99.99=401151]
2018-09-10 14:55:52:298 430 sec: 71642 operations; 718,23 current ops/sec; est completion in 2 minutes [READ: Count=6827,
Max=373759, Min=511, Avg=13449,49, 90=52479, 99=180095, 99.9=328959, 99.99=370175] [UPDATE: Count=356, Max=323839, Min=828
, Avg=15144,94, 90=63615, 99=180735, 99.9=323839, 99.99=323839]
2018-09-10 14:56:02:297 440 sec: 71665 operations; 2,3 current ops/sec; est completion in 2 minutes [READ: Count=22, Max=3
78367, Min=596, Avg=119505,82, 90=378111, 99=378367, 99.9=378367, 99.99=378367] [UPDATE: Count=1, Max=993, Min=993, Avg=99
3, 90=993, 99=993, 99.9=993, 99.99=993]
2018-09-10 14:56:12:297 450 sec: 72514 operations; 84,9 current ops/sec; est completion in 2 minutes [READ: Count=811, Max
=19234815, Min=525, Avg=222276,07, 90=11055, 99=18956287, 99.9=19234815, 99.99=19234815] [UPDATE: Count=38, Max=19234815,
Min=927, Avg=520277,45, 90=64543, 99=19234815, 99.9=19234815, 99.99=19234815]
2018-09-10 14:56:22:304 460 sec: 83302 operations; 1078,15 current ops/sec; est completion in 1 minute [READ: Count=10292,
 Max=341247, Min=519, Avg=9324,98, 90=11767, 99=149247, 99.9=277503, 99.99=338943] [UPDATE: Count=502, Max=210943, Min=797
```

Figure 22 - Exemple d'un workload de mise à jour

Etapes de configuration et d'exécution pour Cassandra :

- Installation et configuration de cassandra, cas d'exemple : 3 nœuds
- Creations de keyspace yscb
- Creation dans ycsb d'une table usertable (y_id, field0, field1, …, field9)

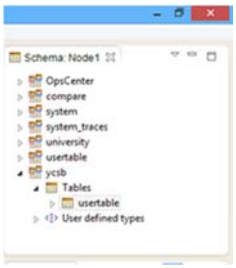

Figure 23 - Keyspace ycsb et table usertable

- Charger les données du workload C dans la base

```
C:\ycsb-0.15.0>bin\ycsb load cassandra-cql -p
hosts=localhost -threads 10 -p fieldcount=10 -p
operationcount=100000 -p recordcount=300000 -P
workloads/workloadd –s > resultat_wlc_l_cassandra.txt
```

- Executer le workload

```
C:\ycsb-0.15.0>bin\ycsb run cassandra-cql -p
hosts=localhost -threads 10 -p fieldcount=10 -p
operationcount=100000 -p recordcount=300000 -P
workloads/workloadd –s > resultat_wlc_r_cassandra.txt
```

- Récupérer les résultats du fichier texte

```
OVERALL], RunTime(ms), 192760
[OVERALL], Throughput(ops/sec), 518.7798298402158
[READ], Operations, 95090
[READ], AverageLatency(us), 18969.628036596907
[READ], MinLatency(us), 376
[READ], MaxLatency(us), 1098751
[READ], 95thPercentileLatency(us), 56607
[READ], 99thPercentileLatency(us), 262399
[READ], Return=OK, 95090
[INSERT], Operations, 4910
[INSERT], AverageLatency(us), 11328.924439918534
[INSERT], MinLatency(us), 373
[INSERT], MaxLatency(us), 38719
[INSERT], 95thPercentileLatency(us), 16103
[INSERT], 99thPercentileLatency(us), 23903
[INSERT], Return=OK, 4910
```

- Passez au workload suivant
- Répétez les mêmes étapes
- Changez la configuration de Cassandra (5 nœuds)

- Répétez les mêmes étapes

Etapes de configuration et d'exécution pour MongoDB :

- Installation et configuration de MongoDB, cas d'exemple : 5 shards
- Creations de database yscb
- Creation dans ycsb d'une collection usertable (y_id, field0, field1, …, field9)

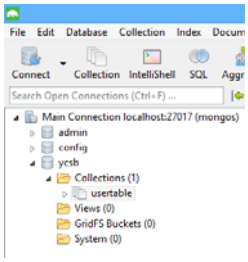

Figure 24 - Database et collection usertable

- Charger les données du workload F dans la base

```
c:\ycsb-015.0>bin\ycsb load mongodb -p
mongodb.url=mongodb://localhost:27017/ycsb -threads 10 -p
fieldcount=10 -p operationcount=100000 -p
recordcount=300000 -p -P workloads/workloadf -s >
wlf_l_mongdb.txt
```

- Executer le workload

```
c:\ycsb-015.0>bin\ycsb run mongodb -p
mongodb.url=mongodb://localhost:27017/ycsb -threads 10 -p
fieldcount=10 -p operationcount=100000 -p
recordcount=300000 -p -P workloads/workloadf -s >
wlf_r_mongdb.txt
```

- Récupérer les résultats du fichier texte

```
[OVERALL], RunTime(ms), 1621672
[OVERALL], Throughput(ops/sec), 61.66475094840387
[READ], Operations, 100000
[READ], AverageLatency(us), 154831.31355
[READ], MinLatency(us), 538
[READ], MaxLatency(us), 51118079
[READ], 95thPercentileLatency(us), 756735
[READ], 99thPercentileLatency(us), 1124351
[READ], Return=OK, 100000
[READ-MODIFY-WRITE], Operations, 50015
[READ-MODIFY-WRITE], AverageLatency(us),
158435.52480255923
[READ-MODIFY-WRITE], MinLatency(us), 1289
[READ-MODIFY-WRITE], MaxLatency(us), 50823167
[READ-MODIFY-WRITE], 95thPercentileLatency(us), 771071
[READ-MODIFY-WRITE], 99thPercentileLatency(us), 1153023
[UPDATE], Operations, 50015
[UPDATE], AverageLatency(us), 5091.211596521043
[UPDATE], MinLatency(us), 658
```

```
[UPDATE], MaxLatency(us), 1423359
[UPDATE], 95thPercentileLatency(us), 3723
[UPDATE], 99thPercentileLatency(us), 82623
[UPDATE], Return=OK, 50015
```

- Pour MongoDB, de préférence de vider la base avant de passer au workload suivant, des fois des valeurs de la clé primaire sont dupliquées.
- Passez au workload suivant
- Répétez les mêmes étapes
- Changez la configuration de MongoDB (10 nœuds)
- Répétez les mêmes étapes

6. Les résultats

Les résultats obtenus de YCSB sont sous forme de chiffres.

```
[OVERALL], RunTime(ms), 609280
[OVERALL], Throughput(ops/sec), 492.3844537815126
[INSERT], Operations, 300000
[INSERT], AverageLatency(us), 20117.620606666667
[INSERT], MinLatency(us), 578
[INSERT], MaxLatency(us), 38567935
[INSERT], 95thPercentileLatency(us), 21071
[INSERT], 99thPercentileLatency(us), 474367
[INSERT], Return=OK, 300000
```

Les résultats les plus intéressants sont :

- Temps d'exécution « Overall, run time » : Le temps nécessaire pour exécuter le workload en milliseconde
- Debit d'operations « Overall, Throughput (Ops/Sec) » :le nombre d'operations par seconde
- Latence de lecture « Read, Average latency » : La latence moyenne de lecture en micro seconde (10^{-6})
- Latence d'écriture « Write, Average latency » : La latence moyenne d'écriture en micro seconde (10^{-6})
- Latence de mise à jour « Update, Average Latency » :La latence moyenne de mise à en micro seconde (10^{-6})jour.

Nous les présentons sous forme de graphes.

7. Analyse des résultats

Les tests devrons passer sur une architecture distribuée Cloud et avec les mêmes paramètres opérationnels (CPUs, RAMs, Disks); vu l'indisponibilité d'un tel environnement commun pour les deux SGBD testés, nous les avons exécuté sur une seule machine avec différentes configurations. Les résultats ne sont pas crédibles à 100%, mais parce qu'ils ont été passés dans les mêmes conditions, nous pouvons comparer les performances des deux SGBD.

Temps d'exécution (Overall Run Time ms)

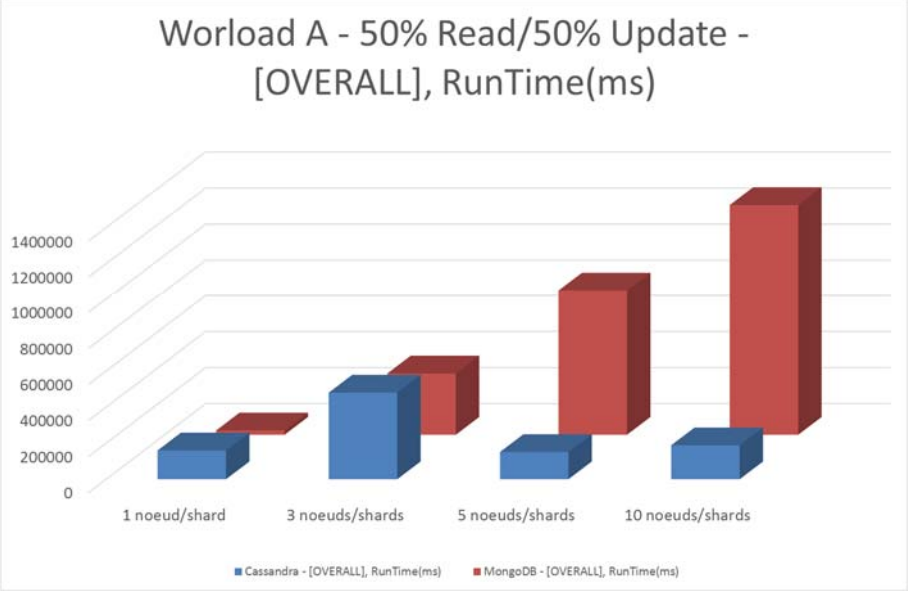

Figure 25 - Workload A, Temps d'exécution

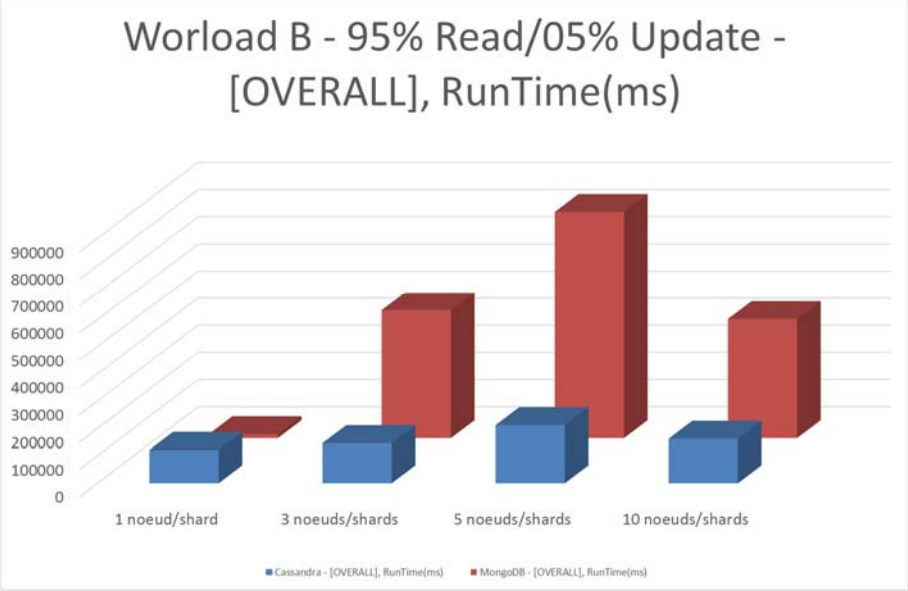

Figure 26 - Workload B, Temps d'exécution

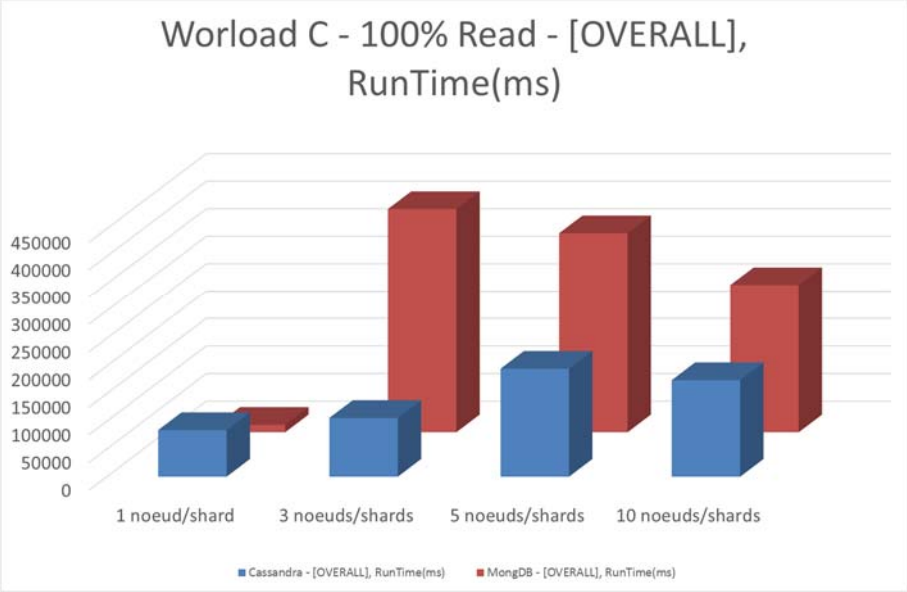

Figure 27 - Workload C, Temps d'exécution

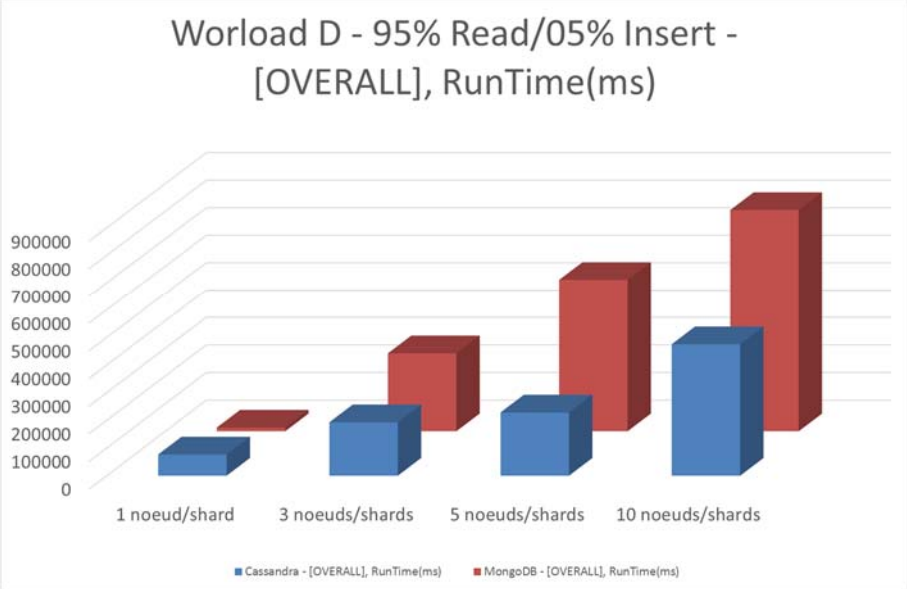

Figure 28 - Workload D, Temps d'exécution

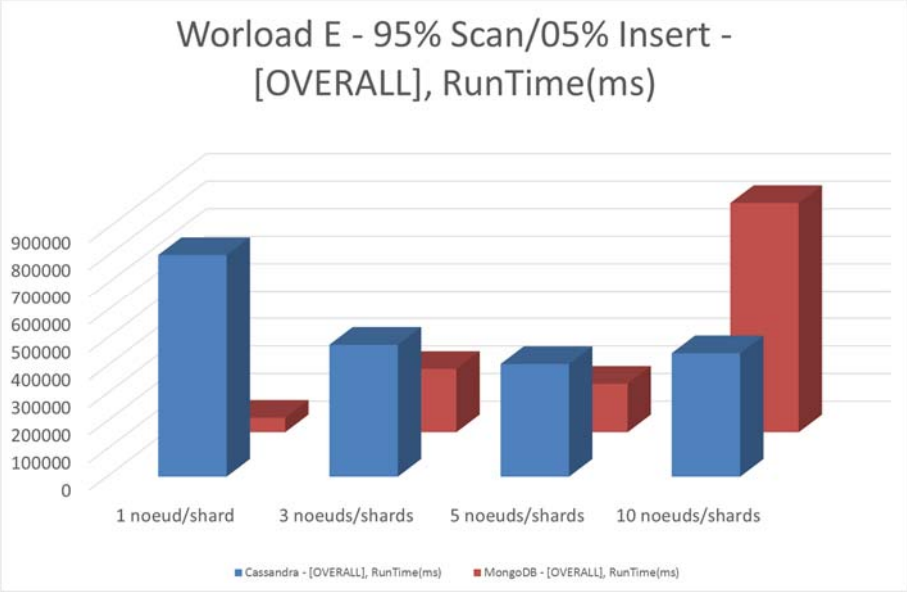

Figure 29 - Workload E, Temps d'exécution

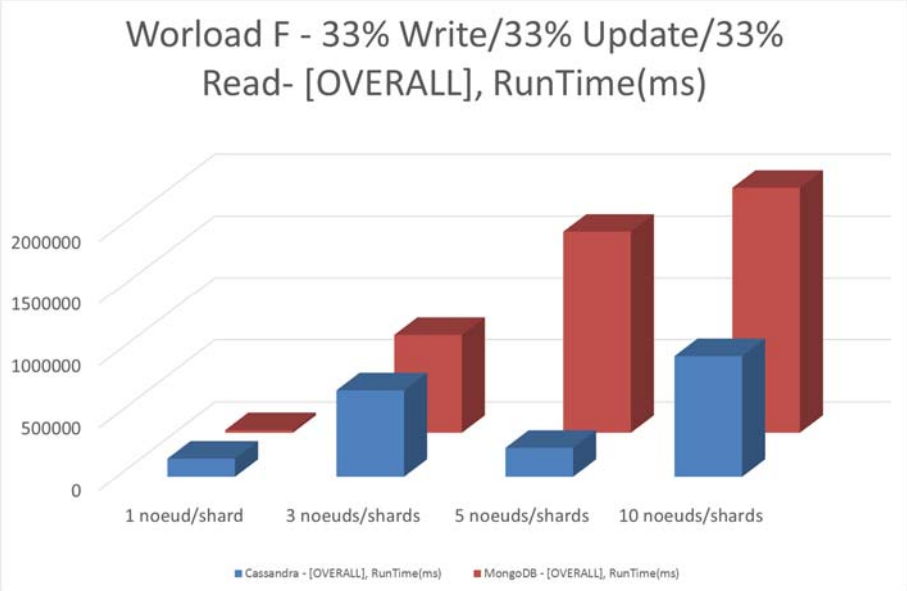

Figure 30 - Workload F, Temps d'exécution

Avec un seul nœud/shard, dans les 6 workloads, le temps d'exécution avec MongoDB est inferieur de celui de Cassandra et même flagrant sur le workload E. au-delà d'un nœud/shard, Cassandra est mieux en temps d'exécution, des fois moins de plus de 50% du temps de MongoDB, le cas du workload A (5 et 10 nœuds), workload B (3, 5 et 10 nœuds), workload C (3 et 5 nœuds), workload D (3, 5 et 10 nœuds), workload E (10 nœuds) et workload F (5 et 10 nœuds).

Débit (nombre d'opérations par seconde)

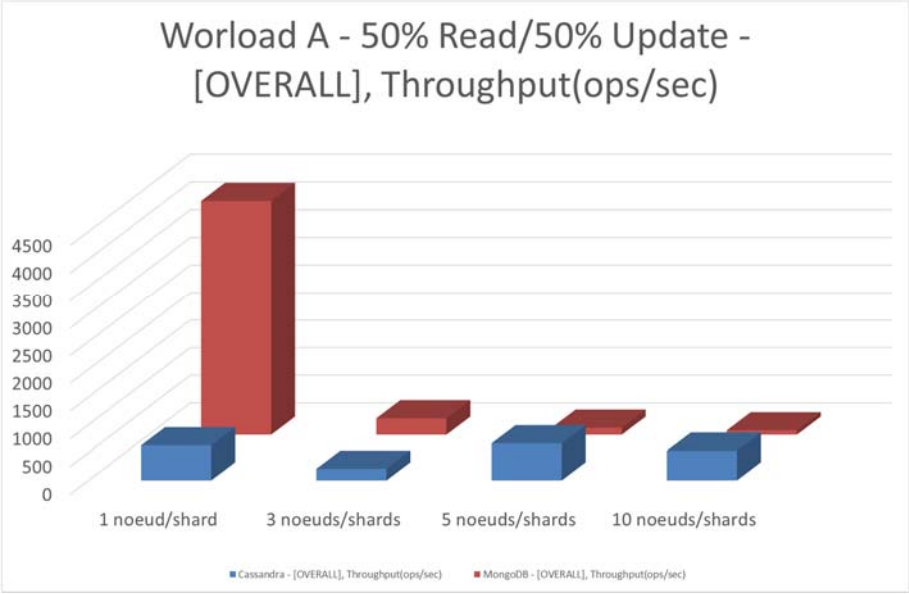

Figure 31 - Workload A, Débit d'opérations

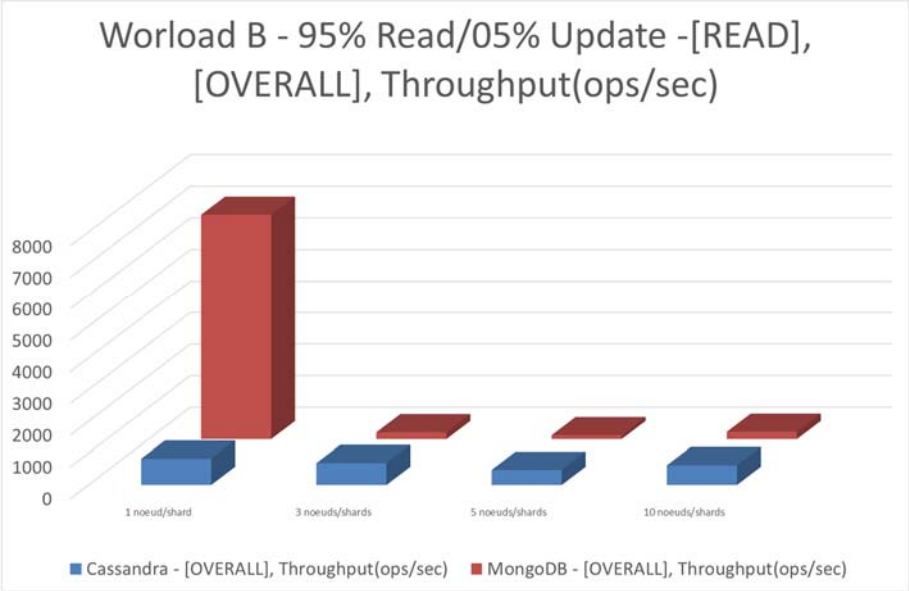

Figure 32 - Workload B, Débit d'opérations

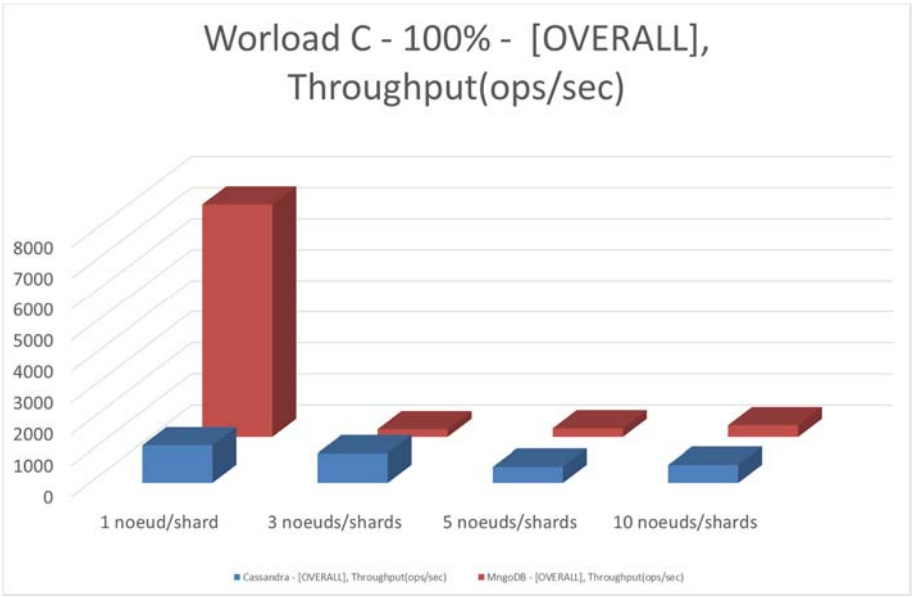

Figure 33 - Workload C, Débit d'opérations

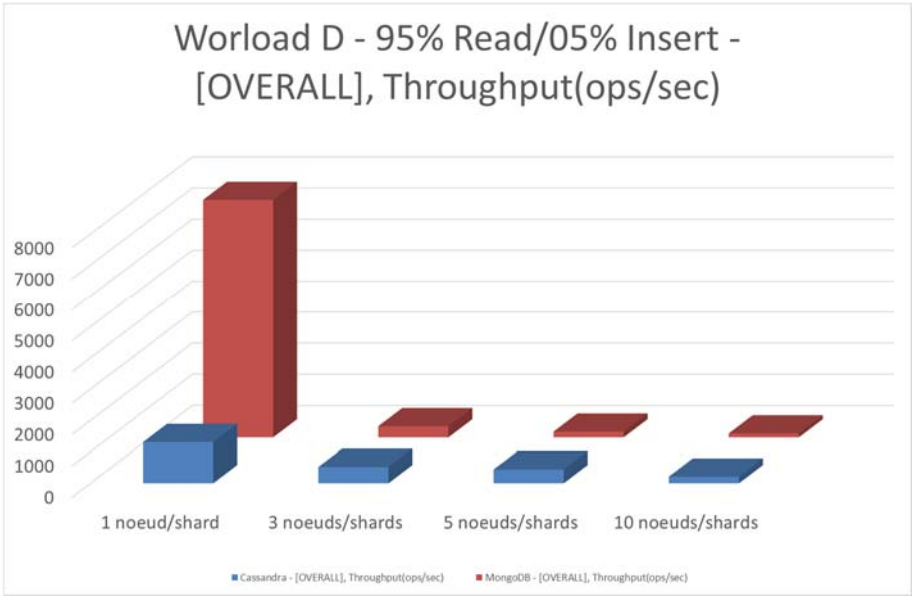

Figure 34 - Workload D, Débit d'opération

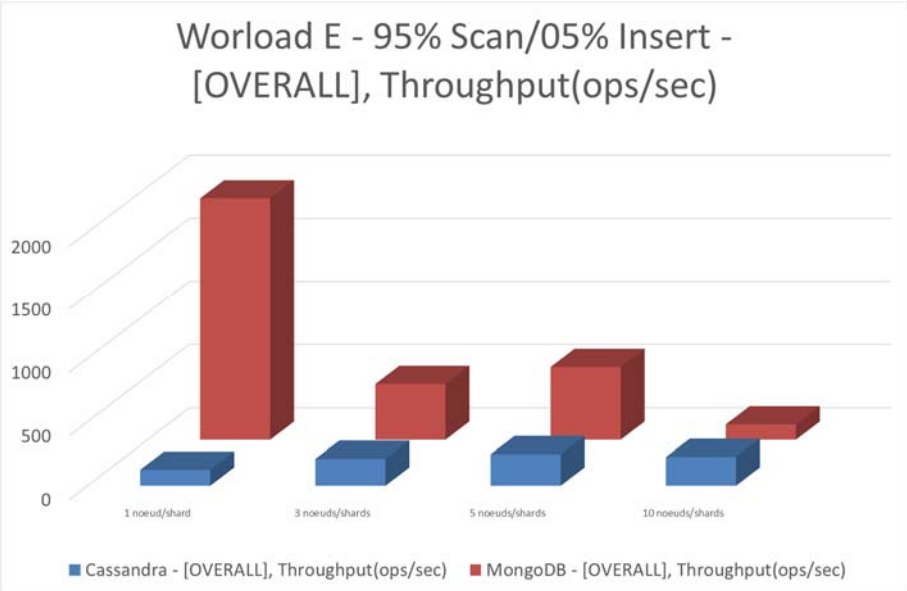

Figure 35 - Workload E, Débit d'opérations

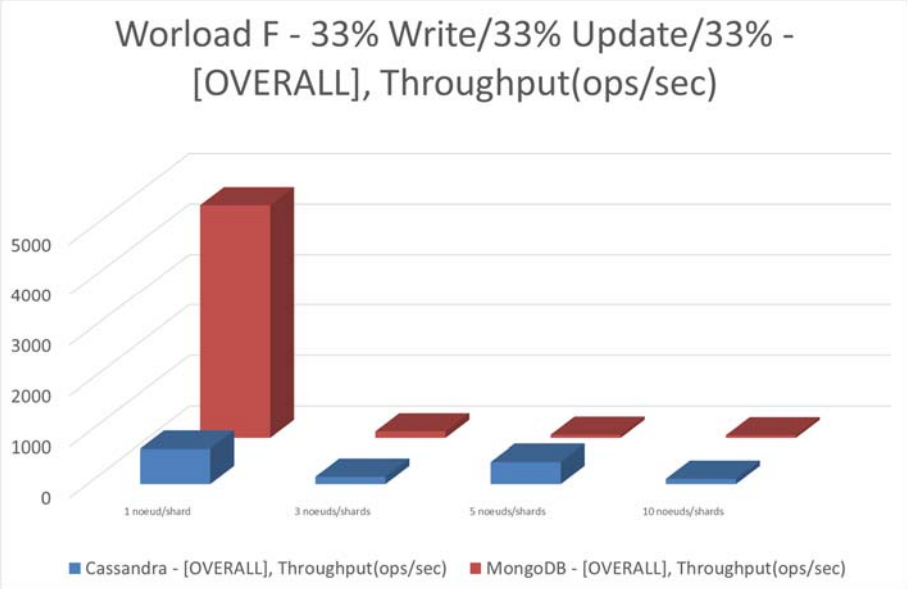

Figure 36- Workload F, Débit d'opérations

Le test du débit sur un seul nœud/shard montrait une différence nette avec les 6 workloads, MongoDB est nettement avancé voire une dizaine de fois quelque fois ; mais à part ça, le débit d'Operations semble presque similaire dans le reste des tests.

Latence de lecture

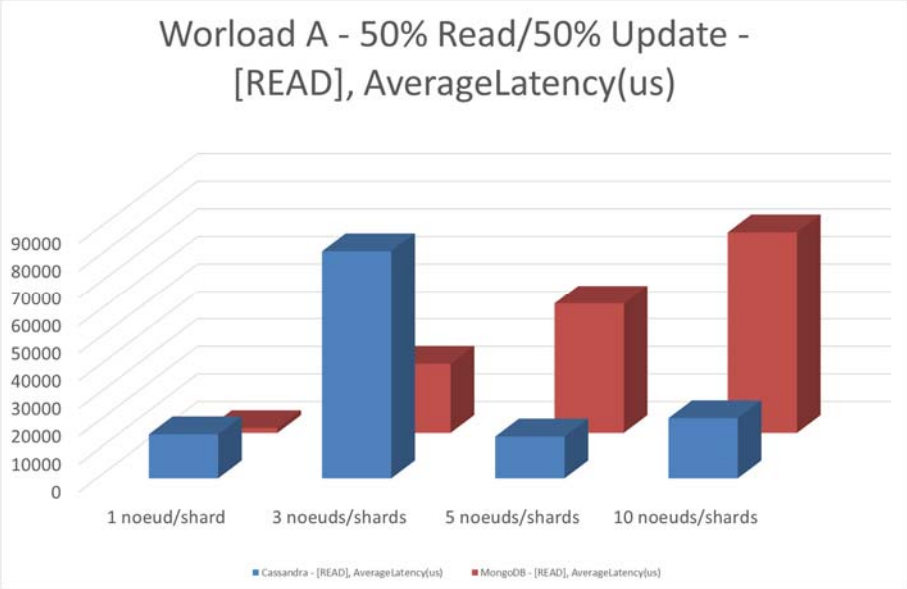

Figure 37 - Workload A, Latence Moyenne de lecture

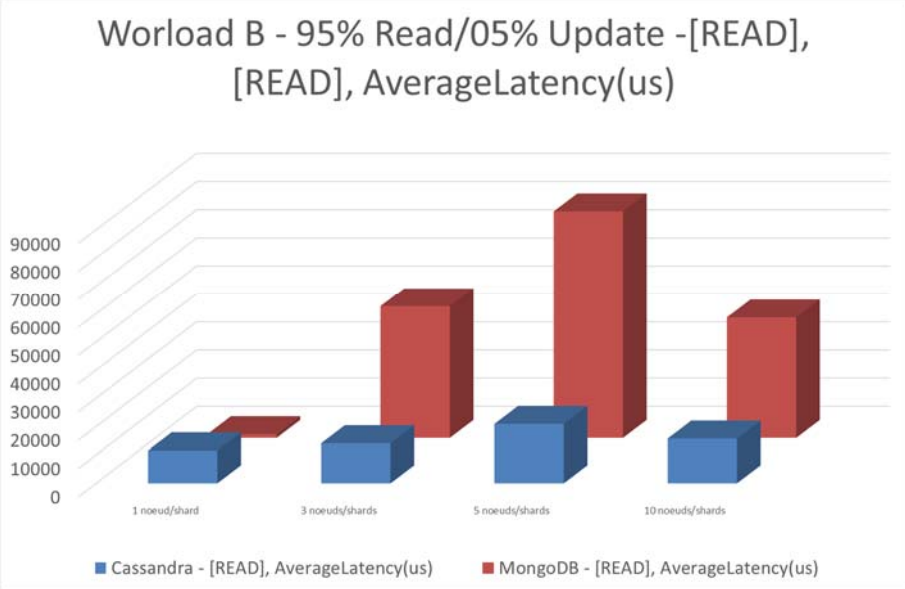

Figure 38 - Workload B, Moyenne de latence de lecture

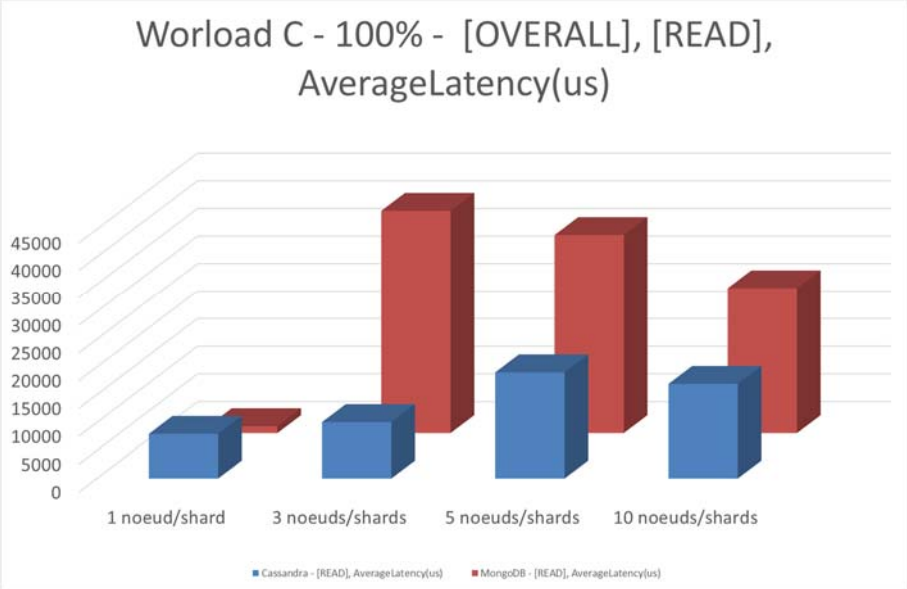

Figure 39 - Workload C, Moyenne de latence de lecture

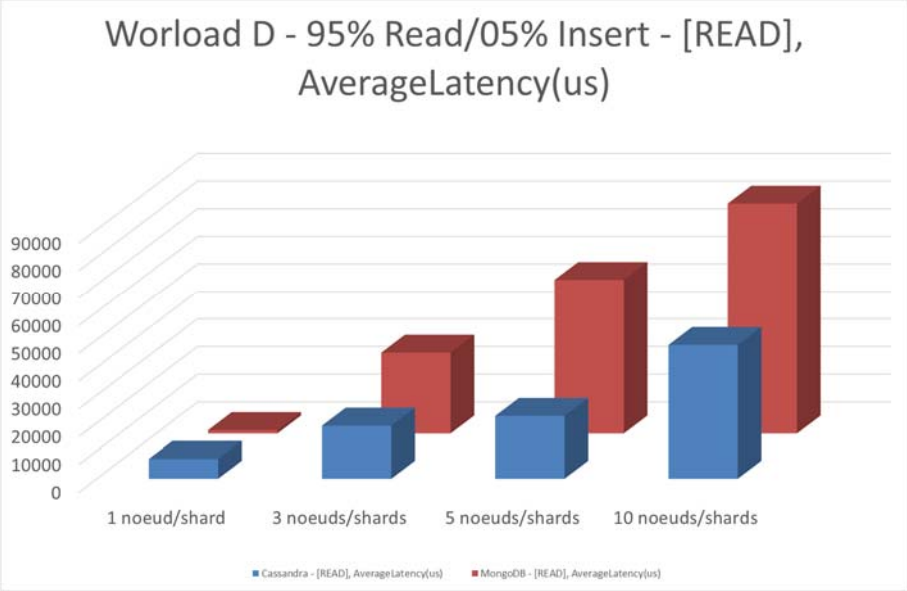

Figure 40 - Workload D, Moyenne de latence de lecture

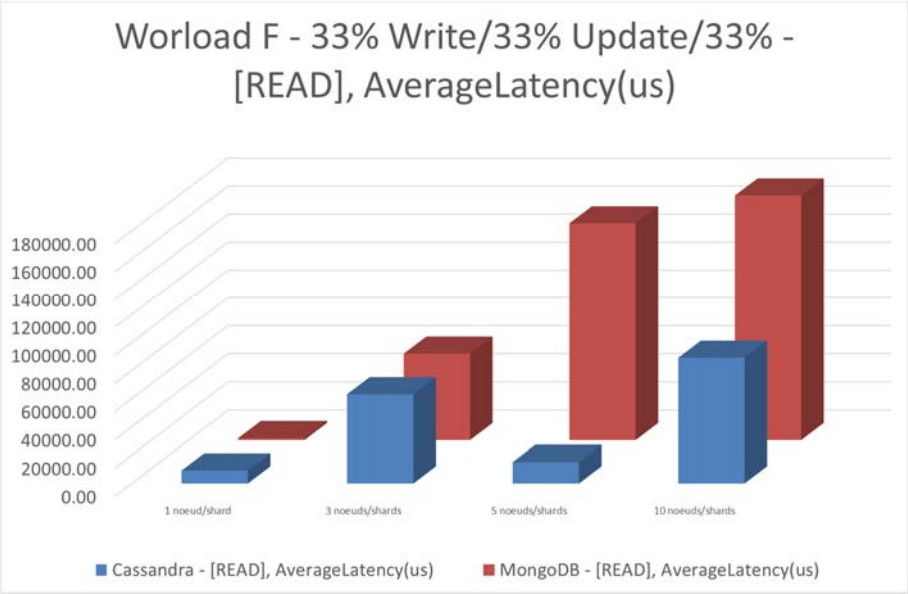

Figure 41 - Workload F, Moyenne de latence de lecture

À part un seul cas (workload A 3 nœuds), dans tous les autres cas, la latence de lecture des enregistrements avec Cassandra est égale ou inférieure à celle de MongoDB.

Latence d'écriture

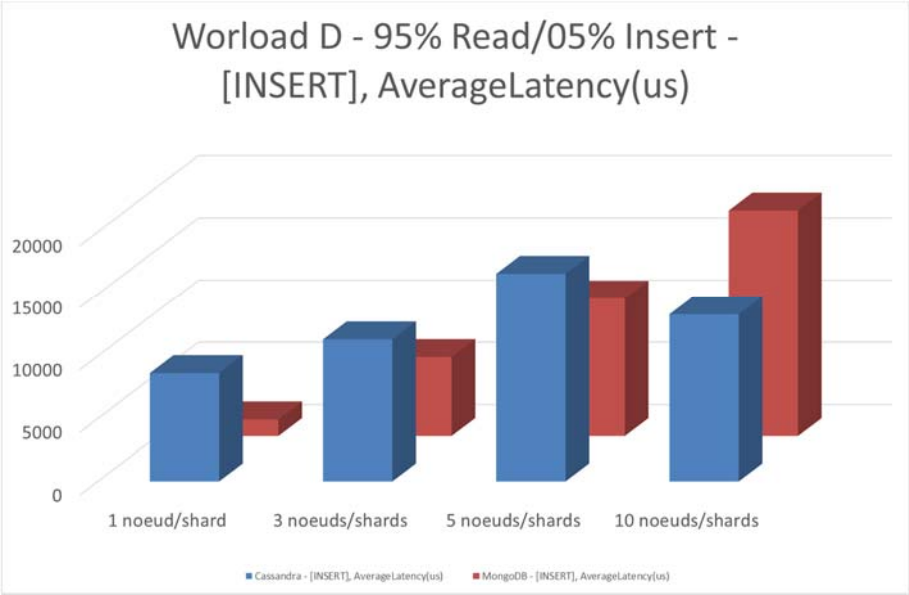

Figure 42 - Workload D, Moyenne de latence d'écriture

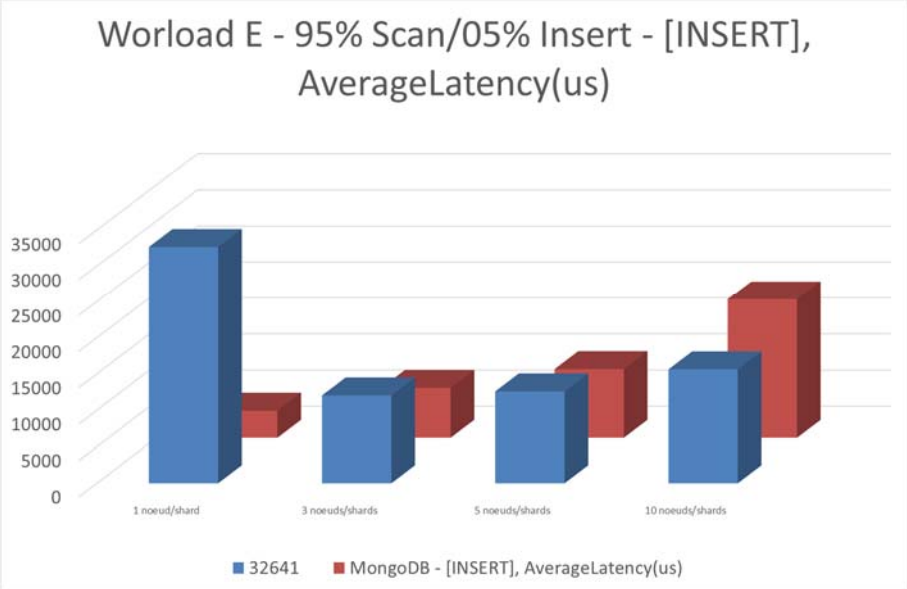

Figure 43 - Workload E, Moyenne de latence d'écriture

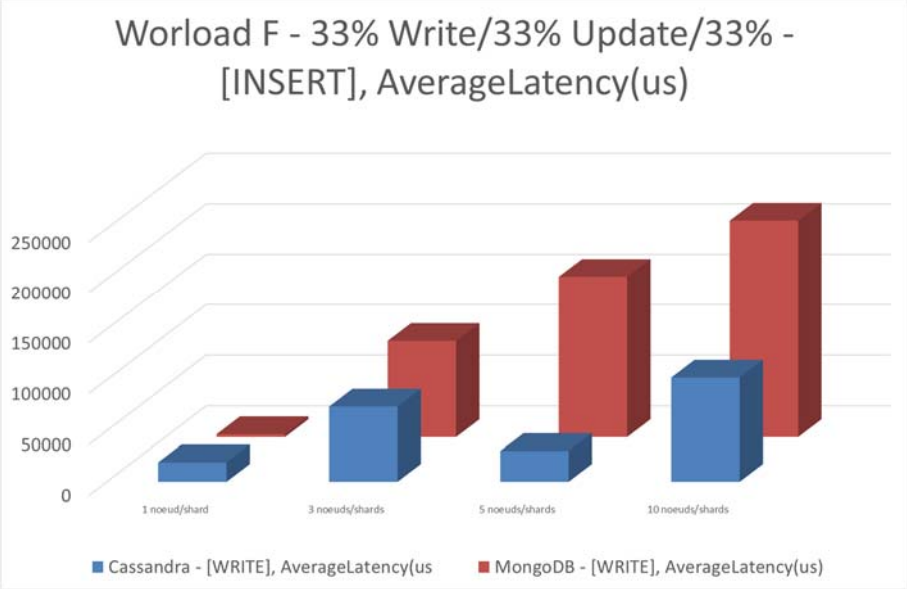

Figure 44 - Workload F, Moyenne de latence d'écriture

La latence d'écriture est testée sur 3/6 des workloads, dans les deux workloads ou l'écriture est légère, 5%, MongoDB est légèrement mieux que Cassandra en général. Dans le workload F, ou l'écriture est de 33% du total des opérations, Cassandra apparait mieux dans la distribution des charges.

Latence de mise à jour

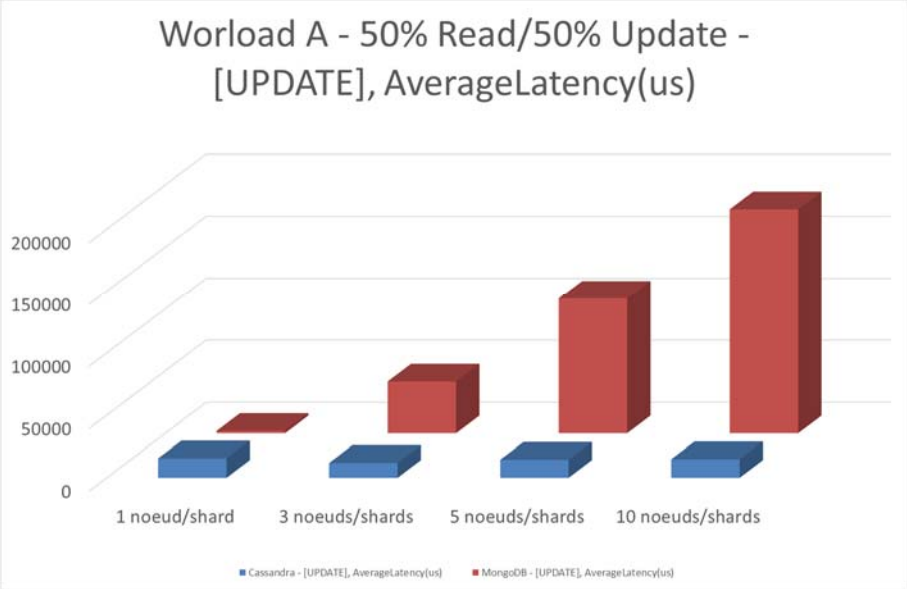

Figure 45 - Workload A, Moyenne de latence de mise à jour

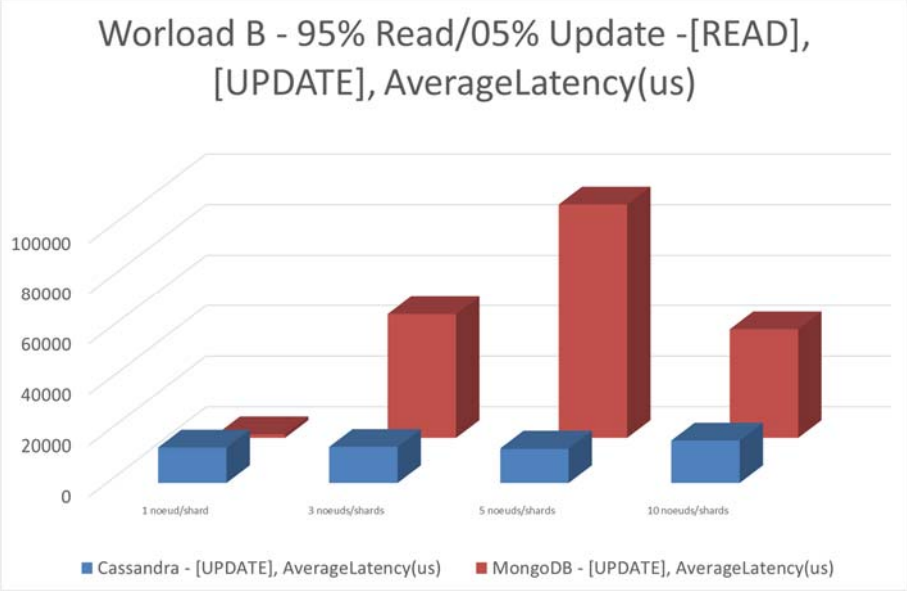

Figure 46 - Workload B, Moyenne de latence de mise à jour

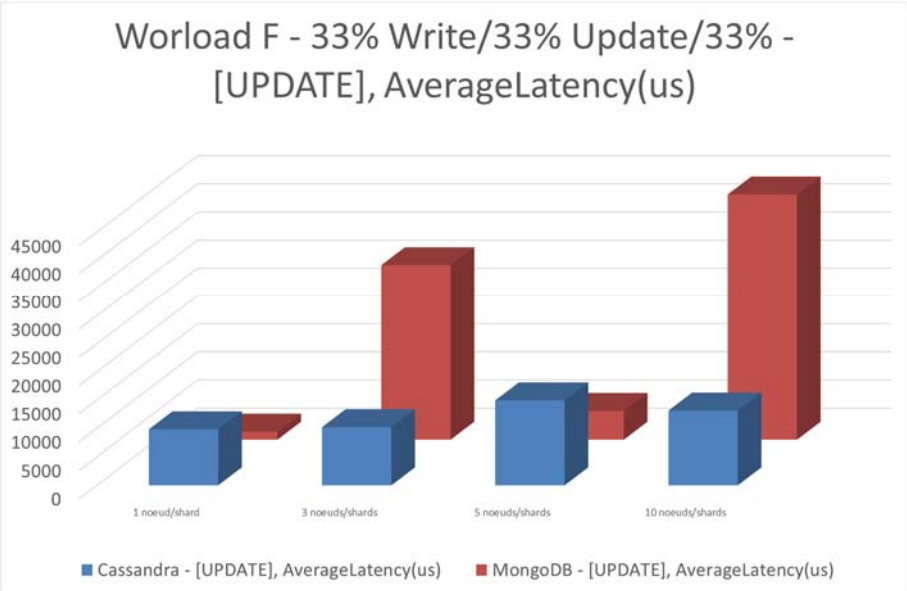

Figure 47 - Workload F, Moyenne de latence de mise à jour

Dans les workloads A et B ou la mise à jour représente 5% et 50% du total des opérations, et à part le test avec un nœud/shard, la latence du Cassandra est meilleure voire incomparable avec celle de MongoDB. Dans le cas où la nature des opérations est distribuée à égalité, le workload F, la latence de mise à jour avec MongoDB est inférieure dans le cas d'un et cinq nœuds et celle de Cassandra dans trois et dix nœuds.

8. Discussion

L'installation des deux SGBDs est facile dépendamment du nombre de nœuds ou de shards voulus, mais la configuration, le déploiement et le maintien de Cassandra est simple, c'est un bon choix pour les systèmes qui s'élargissent exponentiellement. La configuration de MongoDB, replicasets, réplicas primaires, réplicas secondaires, serveurs de données, serveurs de configurations … s'avère compliquée et nécessite le support d'un expérimenté.

Quoique le model de données des 2 SGBD est sans schéma (schemaless), le modèle de MongoDB est plus souple parce qu'il n y a pas de typage de données et il supporte les enregistrements imbriqués. Cassandra est un peu classique, le type de données doit être défini lors de la création des familles de colonnes.

Un avantage de Cassandra est son lagunage de requêtes, le CQL, qui est proche du traditionnel SQL, si vous êtes habitués de SQL, vous n'aurez pas de souci à utiliser le CQL. Par contre, vous devez apprendre les requêtes JSON pour pouvoir interroger les BD MongoDB.

Il est clair que les performances de MongoDB sur un seul shard sont meilleures des performances de Cassandra. Vu que ces systèmes sont destinés à un usage totalement distribué et ont été développés principalement pour pallier l'insuffisance des SGBDR en matière de scalabilité, il ressort clairement que Cassandra supporte bien le passage en échelle en lecture, en écriture et en mise à jour.

La latence d'écriture de Cassandra est nettement inférieure de MongoDB, ceci est jugé par le fait que les réplications de Cassandra sont tous des masters, si un nœud tombe en panne, les écritures basculent automatiquement vers un autre nœud, mais avec MongoDB, ceci requiert entre 10 et 40 secondes pour élire un réplica candidat. Chose qui peut avoir impact sur le critère de la disponibilité. Ainsi, le mécanisme d'écriture de Cassandra favorise sa supériorité parce qu'il utilise la RAM pour stocker ses écritures dans une table appelée mem-table, une fois arrivée à un seuil, les enregistrements sont transférés au disque.

Pour la latence d'écriture, la clé primaire dans Cassandra est obligatoire, cette condition peut être justifie l'avantage de Cassandra. Pour avoir de meilleures performances de lecture avec MongoDB, il est préférable d'utiliser l'indexation.

Pour le temps d'exécution et le débit d'opérations, mais à part l'utilisation d'un nœud, dans tous les tests, les performances de Cassandra sont les meilleures.

Le seul cas ou MongoDB fut meilleurs que Cassandra, était dans le workload E, ou il était demandé de scanner des plages d'enregistrements.

9. Délibération

Comme nous l'avons dit précédemment, ces bases de données sont destinées à une utilisation distribuée, mais par rapport aux résultats obtenus à partir des tests effectués dans le même environnement, nous pouvons conclure que pour des BD rapidement extensibles et qui sont besoin d'ajouter périodiquement des serveurs, avec des taux d'écriture et de mise à jour élevés, Cassandra est la plus appropriée. Citons le cas d'Instagram ; 80 million de photos téléchargées quotidiennement, cet énorme volume de données a besoin d'extension durable.

La base de données (MongoDB) semble être plus appropriée pour les Data Warehouse et les BD analytiques où le taux de scan est critique et les structures de données sont totalement libres.

Conclusion

Notre travail était une étude théorique générale des bases de données NoSQL, leurs définitions, leurs caractéristiques et leurs types ; suivie d'une étude spécifique de deux des plus fameuses sur le marché ; c.à.d. Cassandra et MongoDB ; chacune appartient à une famille différente des BD NoSQL. Nous avons finis par tester leurs performances sur un banc d'essais YCSB et analyser les résultats des tests.

Nous pouvons dire que les BD NoSQL sont la nouvelle génération des SGBD. Ils sont capables de gérer des volumes de données colossales sur un nombre étendu de serveurs avec une haute performance (Youtube, Instagram, Spotify …), leurs marché a atteint les 4 milliard de dollars en 2018. Malgré ce boom, elles ne donnent pas une solution pour toutes les applications. Si les SGBDR répondent à tous nos besoins, il est inutile de migrer vers NoSQL ; exemple, les opérations financières nécessitent la réunion de tous les critères ACID, et surtout une cohérence immédiate est intolérable, ce sont des applications à secrets. Aussi, les applications qui n'ont pas l'intention d'évoluer ou de s'intégrer avec d'autres ; les SGBDR leurs suffit largement. Le NoSQL n'est pas toujours le bon choix. Il est possible de mélanger les deux produits, à savoir le SGBDR et le NoSQL pour avoir le résultat escompté.

Ce modeste travail peut constituer la base d'un travail approfondi à l'avenir, afin d'essayer de tester des bases de données NSQL dans un environnement distribué réel, de préférence Cloud, et tester les cas d'ajout de serveurs avec le système en marche et à l'arrêt, et examiner le cas de défaillance d'un serveur avec système en opération.

VII. Bibliographie

1. *Comparative analysis of NoSQL (MongoDB) with MySQL Database.* **Lokesh Kumar, Dr. Shalini Rajawat, Krati Joshi.** 711, Rajasthan, India : International Journal of Modern Trends in Engineering and Research, 2015, Vol. 1. 2393-8161.

2. **GC, Deepak.** *A Critical Comparison of NOSQL Databases in the Context of Acid and Base.* St. Cloud : Culminating Projects in Information Assurance, 2016. Paper 8.

3. *Comparative Study of SQL and NoSQL Databases to evaluate their suitability for Big Data Application.* **Deepika V. Shetty, Sana J.Chidimar.** 2, Mumbai : International Journal of Computer Science and Information Technology Research, 2016, Vol. 4. 2348-1196.

4. *An advanced comparative study of the most promising NoSQL and NewSQL databases with a multi-criteria analysis method* . **Omar Hajoui, Rachid Dehbi, Mohammed Talea, Zouhair Ibn Batouta.** 03, MOROCCO : International Journal of Computer Science and Information Security, 2018, Vol. 81. 1947-5500.

5. **Garcia-Molina, Hector, Ullman, Jeffery D et Widom, Jennifer.** *Database Systems.* s.l. : Pearson Education, 2009.

6. *A Relational Model of Data for Large Shared Data Banks.* **Codd, Ted.** 1970, Communications of ACM, pp. 377-387.

7. *Towards robust distributed systems.* **Brewer, Eric.** California : ResearchGate, 2000. Symposium on Principles of Distributed Computing. p. 45.

8. **Harrison, Guy.** *Next Generation Databases, .* s.l. : Apress, 2015.

9. **Berman, Jules J.** *Principles of Big Data.* s.l. : Elsevier, 2013.

10. *voyage au coeur du Big Data.* **Clémençon, Stephan.** s.l. : CEA, 2017.

11. **Connolly, Shaun.** 7 key drivers for the big data market. *hortonworks.com.* [En ligne] 14 May 2012. [Citation : 10 April 2018.] https://hortonworks.com.

12. **Vaish, Gaurav.** *Getting Started with NoSQL.* s.l. : Packt, 2013.

13. **David Hows, Eelco Plugge, Peter Membrey, Tim Hawkins.** *The definitive guide to MongoDB.* s.l. : Apress, 2013. 978-1-4302-5822-3.

14. **Russell Bradberry, Eric Lubow.** *Practical Cassandra.* Indiana, USA : Pearson Education Inc, 2014. 978-0-321-93394-2.

15. **Tiwari, Shashank.** *Professional NoSQL.* s.l. : Wrox, 2011.

16. *Dynamo: Amazon's Highly Available Key-value Store.* **DeCandia, Giuseppe, et al., et al.** 2007, Amazon.com, pp. 205-220.

17. *Will NoSQL Databases Live Up to Their Promise?* **Leavitt, Neal.** 10, s.l. : IEEE Computer Society, 2010, IEEE Computer Society, pp. 12-14. 0018-9162.