Comparaison entre Redis et MongoDB

Vincent Boogaart
Département de génie informatique et génie logiciel
Polytechnique Montréal
Montréal, Canada
vincent.boogaart@polymtl.ca

Manel Ben Jemaa
Département de génie informatique et génie logiciel
Polytechnique Montréal
Montréal, Canada
ben-jemaa.manel@polymtl.ca

Henri Larocque
Département de génie informatique et génie logiciel
Polytechnique Montréal
Montréal, Canada
henri.larocque@polymtl.ca

Ahmed Gafsi
Département de génie informatique et génie logiciel
Polytechnique Montréal
Montréal, Canada
ahmed.gafsi@polymtl.ca

Carrie Kam
Département de génie informatique et génie logiciel
Polytechnique Montréal
Montréal, Canada
carrie.kam@polymtl.ca

***Abstract*—L'évolution du paysage informatique a entraîné des défis de gestion des données, passant d'une approche centralisée à une distribution flexible. Cette étude évalue MongoDB et Redis, deux technologies de stockage émergentes, en analysant leurs caractéristiques, performances et applications via YCSB. Chaque base de données est installée sur une instance isolée dans des conteneurs Docker, avec Ubuntu 22.02, 8 Go de RAM et 4 CPU. La configuration comprend une réplication maître-esclave pour les opérations d'écriture, avec trois et cinq nœuds respectivement. Cinq indicateurs sont mesurés: temps d'exécution, débit d'opération, latence de lecture, latence d'écriture et latence de mise à jour, pour chaque configuration et lot de travail.**

Mots clés – base de données, Redis, YCSB, MongoDB, comparaison, temps d'exécution, débit d'opération, latence

I. Introduction

L'évolution du paysage informatique a considérablement modifié la gestion des données, passant d'une approche centralisée à une distribution plus flexible et évolutive. Autrefois, les entreprises se contentaient d'une application unique et d'un matériel physique pour stocker leurs données, mais l'avènement d'Internet a entraîné une explosion des services en ligne et une croissance exponentielle des données à gérer. Cette transition a mis en évidence plusieurs défis, notamment les goulots d'étranglement liés à une surcharge de requêtes sur un seul serveur, les problèmes de performance des applications et la diversité des types de données, rendant impossible une modélisation prédictive complète. [1]

Pour répondre à ces défis, les bases de données distribuées ont émergé comme une solution prometteuse, permettant de partitionner les données et de les stocker sur différents serveurs physiques. Cette approche distribuée offre une scalabilité horizontale et une résilience accrue en répartissant la charge de travail entre plusieurs nœuds, tout en assurant la cohérence et la disponibilité du système. [2] Cette architecture distribuée est également plus économique, permettant une utilisation plus efficace des ressources matérielles. [3]

L'objectif de cette étude est d'évaluer et de comparer deux technologies de stockage de données émergentes: MongoDB et Redis. Nous examinerons en détail leurs caractéristiques, leurs performances et leurs applications respectives, en nous appuyant sur une analyse approfondie de leur documentation et de leurs propriétés intrinsèques. Nous décrirons

également la méthodologie utilisée pour déployer chaque technologie, ainsi que les outils d'analyse employés pour évaluer leurs performances. Enfin, nous comparerons les résultats obtenus pour justifier les différences observées, identifier les limitations et les obstacles rencontrés, et évaluer la pertinence de chaque solution dans des contextes d'utilisation spécifiques.

II. Contexte

A. YCSB

Yahoo propose un outil très puissant appelé YCSB (Yahoo! Cloud Serving Benchmark). Il s'agit d'une nouvelle méthodologie de benchmarking open source, où les utilisateurs peuvent développer leurs propres packages en définissant un nouvel ensemble de paramètres de charge de travail, ou si nécessaire en écrivant du code Java, l'auteur lance un appel pour des benchmarks de scalabilité, suggérant YCSB comme une bonne base de comparaison. Le benchmark mesure les performances brutes, montrant les caractéristiques de latence à mesure que la charge du serveur augmente, et il mesure l'évolutivité, montrant comment le système benchmarké évolue lorsque des serveurs supplémentaires sont ajoutés, et à quelle vitesse le système s'adapte aux serveurs supplémentaires. YCSB se compose de deux composants: un générateur de données et un ensemble de tests de performances pour évaluer les opérations d'insertion, de mise à jour et de suppression des enregistrements. Dans chaque test utilisant certains workloads, nous pouvons configurer le nombre d'enregistrements à charger, le nombre d'opérations à exécuter, et la proportion de lecture et d'écriture, par exemple: Workload A: 50% de lectures, 50% de mises à jour). Ces workloads peuvent être modifiés et personnalisés en fonction du type de résultats de test attendus. [4]

B. MangoDB

MongoDB est une base de données non relationnelle open source qui stocke les données sous format binaire de type JSON, appelé BSON et offre des opérations atomiques sur les champs. [5] Elle propose deux méthodes de réplication: maître-esclave et ensembles de répliques. Dans le cadre de la réplication maître-esclave, le maître gère l'accès complet aux données et envoie les modifications aux esclaves, qui ne peuvent que lire les données. Quant aux ensembles de répliques, le concept est similaire à la réplication maître-esclave, mais un nouveau maître peut être élu en cas de panne du maître d'origine. Une autre fonctionnalité clé offerte par MongoDB est le partitionnement automatique, également appelé *sharding*, permettant de répartir les données sur différents nœuds. [6] Chaque collection doit avoir une clé de partitionnement définie par l'administrateur, ce qui permet de déterminer comment les documents sont répartis, ce qui le rend efficace pour les requêtes complexes. [5] Lorsque les clients se connectent à un nœud maître, appelé processus mongo, celui-ci analyse et redirige la requête vers le ou les nœuds appropriés.

MongoDB est une base de données orientée document reconnu pour sa flexibilité de schéma, sa facilité de scalabilité horizontale et sa riche langue de requête.

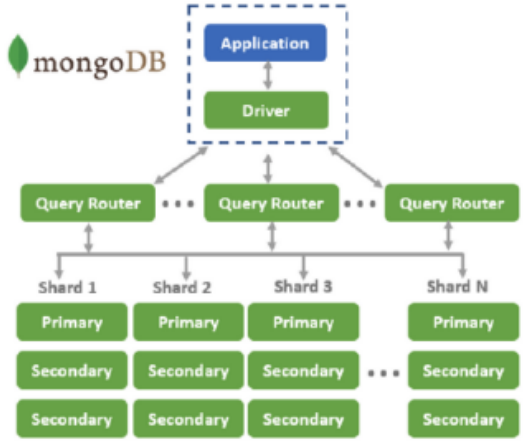

Fig. 1. Architecture MangoDB [7]

C. Redis

Redis est une base de données clé-valeur en mémoire, offrant une variété de structures de données telles que des chaînes, des listes, des ensembles, des ensembles ordonnés et des hachages. Redis stocke toutes les données en mémoire et les écrits périodiquement sur le disque. Bien que Redis propose une réplication maître-esclave, il ne prend pas en charge les requêtes complexes ni l'indexation. Malgré cette limitation, Redis est réputé pour sa simplicité et sa rapidité, en particulier pour les opérations de base clés-valeurs. Cette

combinaison de vitesses et de structures de données simples en fait l'une des implémentations KVS les plus rapides pour de nombreuses applications, idéale pour le coaching, la gestion de session et les applications en temps réel. Cependant, il est important de noter que sa capacité à répondre aux besoins de stockage et de traitement des données peut être limitée dans les cas nécessitant une manipulation complexe des données ou des opérations d'indexation avancées. [8]

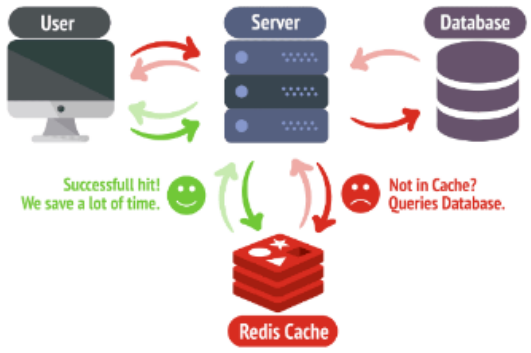

Fig. 2. Architecture Redis [9]

D. Comparaison

En résumé, Redis est une base de données en mémoire, optimisée pour des opérations clé-valeur rapides, tandis que MongoDB est une base de données documentaire utilisant JSON pour stocker les données de manière persistante. Les deux offrent une évolutivité horizontale, mais diffèrent dans leur modèle de données, leur intégrité, leur langue de requête et leurs capacités de mise à l'échelle. Le choix dépend des besoins spécifiques de l'application. [10] Voici un tableau qui compare les deux bases de données:

TABLE I
TABLEAU DE COMPARAISON

Critères	**MongoDB**	**Redis**
Base de données	Utilisé comme base de données principale [11]	Utilisé comme base de données en cache [11]
Modèle de données	BSON (Binary JSON) [11]	N/A (Redis n'a pas besoin de formatage spécifique) [11]
Mémoire clé-valeur	N/A (MongoDB ne fonctionne pas avec une mémoire clé-valeur) [11]	Utilisé comme une mémoire clé-valeur rapide et efficace [11]
Stockage en données	Stocke les données principalement sur un disque [10]	Utilise principalement la RAM [10] pour le stockage en cache [12]
Mise en échelle	Utilise le partitionnement pour répartir les données [10]	Utilise le partitionnement, mais avec une seule partition par défaut [10]
Disponibilité	Offre la disponibilité par la réplication [10]	Offre la disponibilité par la réplication, mais il a besoin de Redis Sentiel pour le basculement automatique [10]
Architecture	Master slave peer to peer via sharding [13]	Master slave architecture [13]
Domaine de l'utilisation	Système CMS, stockage [13]	Géospatial, média riche streaming, temps réel analytique, machine et apprentissage [13]

III. Revue de littérature

A. Redis

L'article fournit une vue d'ensemble des différentes structures de données offertes par Redis, ainsi que des modules supplémentaires qui améliorent ses fonctionnalités de base. Redis se distingue des autres bases de données NoSQL et de stockage clé-valeur par sa variété de structures de données, notamment les chaînes, les hachages, les ensembles ordonnés, les ensembles non ordonnés et les listes [14].

Les chaînes, qui peuvent stocker divers types de données telles que du texte, des entiers, des décimales, des images, des vidéos ou des fichiers audio, sont omniprésentes dans toutes les bases de données clé-valeur. Elles offrent des opérations telles que l'incrémentation de valeurs numériques et la manipulation de chaînes. [14]

Les hachages permettent de stocker de multiples valeurs dans un objet compact, ce qui est utile pour regrouper des informations, par exemple les données des étudiants. [14]

Les ensembles ordonnés sont particulièrement utiles pour obtenir des données dans un ordre spécifique, ce qui facilite la recherche d'articles les plus consultés, de commandes par montant, de scores élevés, etc. [14]

Les ensembles non ordonnés et les listes permettent respectivement de stocker des éléments sans ordre particulier et des éléments liés les uns aux autres, offrant des fonctionnalités telles que l'intersection, la différence et l'union pour les en-

sembles, ainsi que des opérations d'insertion pour les listes. [14]

En outre, Redis est enrichi par des modules supplémentaires qui étendent ses fonctionnalités de base. Par exemple, RediSearch fournit une recherche full texte puissante, RedisJSON permet de manipuler des documents JSON de manière efficace, RedisGears facilite la combinaison de structures de données multiples, RedisAI fournit des modèles de deep learning et de machine learning hautement disponible et évolutif, RedisGraph offre une base de données de graphes rapide, et RedisTimeSeries simplifie le stockage et la manipulation de données temporelles. [14].

En conclusion, Redis se démarque par sa diversité de structures de données et par les modules supplémentaires qui étendent ses capacités, ce qui en fait un choix attrayant pour une variété d'applications de base de données.

B. MangoDB

D'après une expérimentation avec quatre plateformes Blockchain (Hyperledger Fabric, Hyperledger Burrow, Hyperledger Sawtooth et BigchainDB) et une configuration de MongoDB, MongoDB n'est pas stable lorsqu'il y a un changement abrupt de taille de réseau. Le réseau simulé représente trois bâtiments intelligents contenant des dispositifs IoT [15].

Dans une autre expérimentation entrent MongoDB et Cassandra, les auteurs ont constaté qu'il y a une diminution de performance (augmentation de latence en lecture et en mise à jour, puis une diminution de débit) quand on augmente le nombre de threads. L'expérimentation est faite avec 1 Go de données générées par YCSB [16].

C. Comparaison des performances

Pour Redis, selon l'article de Nadia Ben Seghier et Okba Kazar [13], ses performances supérieures lors des opérations de lecture sont attribuées à son utilisation de la mémoire volatile pour le stockage et la récupération des données. De plus, il se distingue aussi par sa rapidité dans les opérations d'écriture. [13]

Concernant MongoDB, il se démarque par un temps de chargement plus rapide. Pour l'expérimentation, les auteurs ont utilisé cet environnement de test avec l'outil YCSB: [13]

Hardware	Software
Processor : Intel® Core™ i3-2330M CPU @ 2.20 GHz RAM : 4.00 GB HDD: 450 GB	Ubuntu 18.04 , PC 64-bit YCSB 0.18.0 Redis 4.0.9 MongoDB 3.6.3 Cassandra 3.11.6

Fig. 3. Environnement de test. [13]

Selon l'expérience menée par Yaser Mansouri, Victor Prokhorenko, Faheem Ullah et Muhammad Ali Babar [17], Redis est plus économe en énergie lorsque des nœuds périphériques, provenant des ordinateurs portables ou des grappes de Raspberry Pis, disposent d'une grande capacité de mémoire. Dans tous les cas, la consommation de Redis est inférieure à MongoDB, à l'exception du cas d'où c'est Raspberry Pi vers le nœud de bordure via Wifi. En général, sa performance en termes de consommation d'énergie diminue, lorsqu'il est utilisé localement. En revanche, lors du transfert de données de la base de données vers un nuage hybride (OpenStack et Azure), Redis consomme le moins d'énergie, ce qui est également influencé par le grand nombre de nœuds dans le nuage public.

Dans nos revues de littérature, les expérimentations sont toutes différentes de ce que nous avons, à l'exception d'une. Donc, nous avons des données venant avec des configurations et des paramètres de tests différents. Les revues de littérature explorées pour cette étude offrent un aperçu des performances et des capacités des systèmes de gestion. Certaines revues de littératures ont des points communs et des différences qui ont été identifiés, ce qui peut orienter nos investigations. En général, Redis se distingue par sa diversité de structures de données telles que des chaînes, des hachages, des ensembles ordonnés et non ordonnés, ainsi que des listes, enrichies par des modules supplémentaires qui étendent ses capacités. Aussi, sa capacité à exécuter rapidement les opérations de lecture et d'écriture, grâce à l'utilisation de la mémoire volatile, le rend particulièrement intéressant pour des applications nécessitant de hautes performances en temps réel. Concernant MongoDB, malgré ses temps de chargement rapides, des instabilités ont été observées lors de changements abrupts de

taille de réseau et une diminution de performance avec l'augmentation du nombre de threads. Ces observations soulignent des points de vigilance particuliers pour des environnements exigeant une haute stabilité et scalabilité. De plus, une étude comparative sur l'efficacité énergétique a montré que Redis est généralement plus économe, surtout lorsque les nœuds périphériques disposent d'une grande capacité de mémoire, bien que cette performance varie selon l'environnement, notamment lors du transfert de données vers des nuages hybrides. Toutefois, cette efficacité varie selon les environnements, notamment lors de transferts de données vers des nuages hybrides, suggère que les conditions spécifiques d'utilisation doivent être évaluées.

Pour nos futures investigations, il est essentiel de conduire des tests dans des conditions réalistes, en incluant un éventail plus large d'opérations pour mieux simuler les charges de travail réelles. Cela permettra non seulement de confirmer ou de réfuter les conclusions des études précédentes, mais aussi de fournir des données plus représentatives des performances de MongoDB et Redis dans des scénarios d'utilisation quotidiens. En résumé, selon les critères de notre expérimentation, les revues de littératures nous disent que Redis est plus préférable que MongoDB.

IV. Évaluation des performances

Après avoir effectué une analyse approfondie du sujet et présenté toutes les informations concernant le contexte de notre expérience, nous nous concentrerons sur la seconde partie, qui est consacrée à la pratique. Plus précisément, nous allons évaluer les performances de deux outils que nous avons choisis pour notre étude: MongoDB et Redis pour le système de gestion de base de données NoSQL.

A. Hypothèses

D'après les informations fournies, on peut déduire que Redis et MongoDB présentent des différences significatives en termes de performances de lecture et d'écriture.

Pour Redis, il excelle dans les opérations de lecture en raison de son utilisation de la mémoire volatile pour le stockage et la récupération des données. De plus, Redis est réputé pour sa rapidité dans les opérations d'écriture, ce qui en fait une option attrayante pour les applications nécessitant des opérations clé-valeur rapides [8]. Également, il est noté que Redis consomme généralement moins d'énergie que MongoDB dans la plupart des cas, ce qui pourrait avoir un impact sur la vitesse de lecture et d'écriture [17].

En ce qui concerne MongoDB, il est souligné un temps de chargement plus rapide, ce qui peut être un avantage dans certaines situations [13].

Dans l'ensemble, ces informations suggèrent que Redis pourrait être préférable pour des applications nécessitant des opérations rapides de lecture et d'écriture, tandis que MongoDB pourrait être plus adapté pour des scénarios nécessitant un temps de chargement rapide. Les résultats des expérimentations à venir permettront probablement de confirmer ces hypothèses et de fournir des perspectives plus détaillées sur les performances et les fonctionnalités de chaque base de données.

B. Expérimentations

1) Configuration globale: Quelques outils ont été utilisés pour mettre en place notre infrastructure de test. Tout d'abord, nous utilisons docker-compose, qui nous permet de configurer et de coordonner facilement un ensemble de conteneurs. Ensuite, en ce qui concerne l'infrastructure physique, nous utilisons un environnement Linux basé sur la distribution Ubuntu 22.04 LTS, avec une utilisation maximale de 8 gigaoctets (Go) de mémoire vive.

2) Paramètres d'expérimentation: Nous avons expérimenté avec deux types de configurations de clusters de bases de données : l'une à trois nœuds et l'autre à cinq nœuds. Dans ce contexte, un nœud représente une instance de la base de données Redis. Les configurations utilisent la stratégie de réplication Maître-Esclave. Ainsi, chaque configuration possède un nœud maître, qui sert de point d'entrée de la base de données. Les autres nœuds sont simplement des répliques, c'est-à-dire des copies du maître. La configuration à trois nœuds comprend donc deux répliques, tandis que celle a cinq nœuds en compte quatre répliques.

Du côté de MongoDB, la configuration est très similaire, l'unique différence étant que les nœuds sont référés en tant que nœuds primaires et secondaires plutôt que maîtres et esclaves. La

configuration est définie dans un fichier Docker Compose où chacun des conteneurs de base de données partage le même Replica Set (ensemble de réplication) et où le nœud primaire exécute une commande pour initialiser le Replica Set avec les deux autres nœuds. La version 5.0.26 de MongoDB a été utilisée afin d'assurer la compatibilité avec l'outil YCSB. Tout comme pour Redis, on a une configuration de trois nœuds ainsi qu'une de cinq nœuds pour MongoDB.

Pour exécuter et automatiser nos tests de charge sur ces configurations, nous avons utilisé un ensemble de scripts Shell et Python. Le premier script Shell se charge de lancer et détruire l'environnement Docker-Compose après chaque ronde de test. Le deuxième script orchestre l'ensemble des 10 rondes de tests par configuration. Quant au script Python, celui-ci utilise l'outil de ligne de commande YSCB, qui exécute un test de performance selon un ensemble de paramètres fournis. Dans le cas de MongoDB, le pilote asynchrone a été utilisé. Chaque ronde de test exécutée par ce script Python est caractérisée par trois tests de performance, variant en fonction de leurs paramètres. Les détails de ces tests seront abordés dans la section suivante.

3) Configuration des lots de tests: Comme mentionné dans la section précédente, pour nos expérimentations, nous avons employé un ensemble de trois lots de travail. Ceux-ci partagent certaines caractéristiques et différents d'autres. Pour les paramètres partagés, nous avons spécifié un ensemble de 10 000 records, c'est-à-dire un item dans la base de données. Cela signifie que la base de données sera déjà peuplée avec un nombre important d'item, ce qui permet de simuler un état réel. Puis, nous avons spécifié une limite d'opérations, soit le nombre d'opérations qu'un test a réalisées. Nous avons placé cette limite à dix mille. De l'autre côté, dans les configurations différentes, il s'agit de la proportion de types d'opération. Nous avons un lot de travail de lecture, qui possède seulement des opérations de lecture. Puis, nous avons un lot de travail équilibré dont la moitié des opérations sont des opérations de lecture et l'autre moitié sont d'écriture. Finalement, nous avons un lot de travail d'écriture qui possède à 90% des opérations d'écriture et à 10% des opérations de lecture. Pour les opérations d'écriture, nous les avons séparées également entre des opérations d'insertions et des opérations de mises à jour, afin de pouvoir visualiser si une différence de performance existait pour ces deux types d'opérations. Voici un tableau qui résume visuellement ce paragraphe:

TABLE II
TABLEAU DES WORKLOADS

Lot de travail	Paramètre partagé	Nature de l'opération	Pourcentage
Workload 100/0	10 000 records 10 000 opérations	Lecture	100
		Mise à Jour	0
		Insertion	0
Workload 50/50	10 000 records 10 000 opérations	Lecture	50
		Mise à Jour	25
		Insertion	25
Workload 10/90	10 000 records 10 000 opérations	Lecture	10
		Mise à Jour	45
		Insertion	45

4) Résultats des tests: Pour chacun de nos lots de tests définis, nous avons collecté un ensemble de données utile à l'analyse de la performance de Redis et de MongoDB: la latence moyenne pour les opérations d'écriture, de lecture et de mise à jour, le rendement moyen, soit le nombre d'opérations moyen effectué par seconde et le temps total pris pour effectuer le lot de travail.

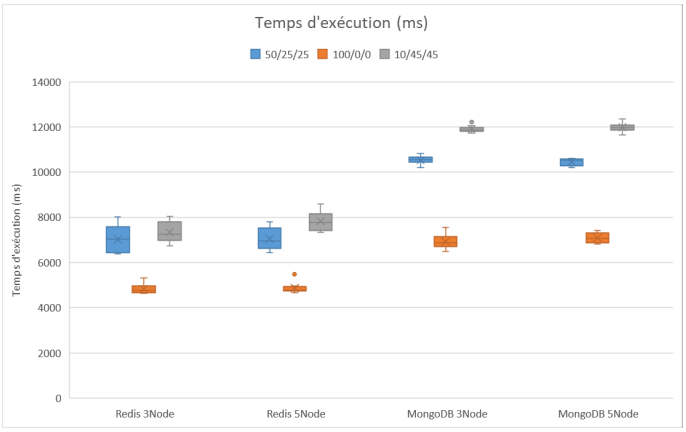

Fig. 4. Temps d'exécution (ms)

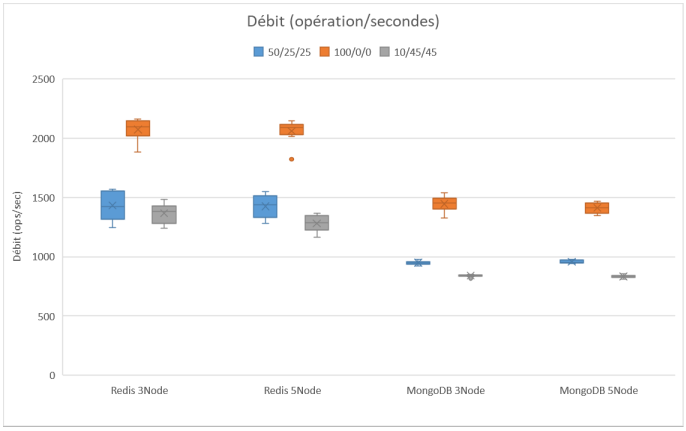

Fig. 5. Débit (opération/secondes)

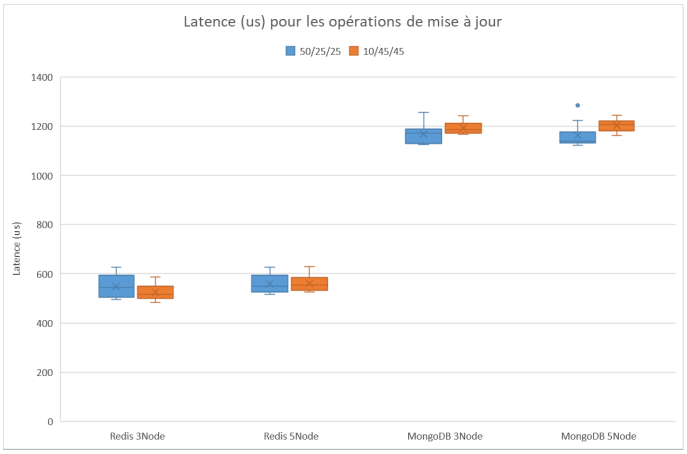

Fig. 8. Latence (μs) pour les opérations de mise à jour

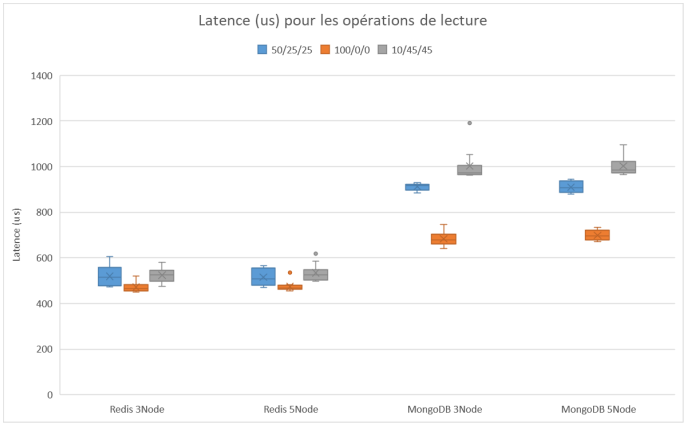

Fig. 6. Latence (μs) pour les opérations de lecture

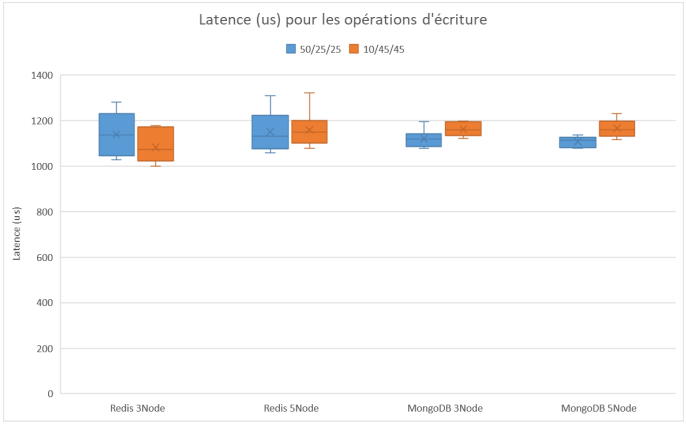

Fig. 7. Latence (μs) pour les opérations d'écriture

TABLE III
TEMPS D'EXÉCUTION TOTAL (MS)

Configuration	50-50	100-0	10-90
Redis 3 nodes	7179	4820	7413
Redis 5 nodes	6994	4833	7867
MongoDB 3 nodes	10577	6884	11925
MongoDB 5 nodes	10406	7102	12019

TABLE IV
DÉBIT GLOBAL (OPÉRATIONS/SEC)

Configuration	50-50	100-0	10-90
Redis 3 nodes	1404	2106	1357
Redis 5 nodes	1441	2090	1277
MongoDB 3 nodes	953	1472	840
MongoDB 5 nodes	960	1419	831

TABLE V
LATENCE MOYENNE EN LECTURE (µS)

Configuration	50-50	100-0	10-90
Redis 3 nodes	522	458	529
Redis 5 nodes	512	465	552
MongoDB 3 nodes	911	692	980
MongoDB 5 nodes	921	707	985

TABLE VI
LATENCE MOYENNE EN INSERTION (µS)

Configuration	50-50	100-0	10-90
Redis 3 nodes	1154	1008	1130
Redis 5 nodes	1145	1103	1251
MongoDB 3 nodes	1120	1079	1139
MongoDB 5 nodes	1115	1092	1156

TABLE VII
LATENCE MOYENNE EN MISE À JOUR (µS)

Configuration	50-50	100-0	10-90
Redis 3 nodes	553	500	523
Redis 5 nodes	571	529	564
MongoDB 3 nodes	1166	1124	1190
MongoDB 5 nodes	1152	1132	1203

5) Description des résultats: Les résultats obtenus à partir des graphiques de performance révèlent une supériorité marquée de Redis en termes de temps d'exécution, de débit des opérations, de latence pour la lecture et de latence pour la mise à jour par rapport à MongoDB, ce qui est constant à travers les différentes configurations de cluster. Redis excelle dans les opérations de lecture, mettant en évidence l'avantage de son stockage en mémoire pour un accès rapide aux données. Le nombre de nœuds dans le cluster n'a pas d'impact prononcé sur les performances de MongoDB (excepté pour les opérations de lecture et de mise de à jour), ce qui pourrait indiquer des limites dans l'efficacité de sa mise à l'échelle dans le cadre de ces tests. Pour la latence des opérations d'écriture, Redis et MongoDB sont similaires au niveau des valeurs. Ces constats confirment l'importance des choix architecturaux sous-jacents dans la performance des bases de données et suggèrent que Redis pourrait être préféré pour des opérations nécessitant de la rapidité et de la réactivité, tandis que MongoDB pourrait être mieux adapté aux applications où les opérations complexes et la persistance des données sont critiques.

C. Discussion

L'analyse des résultats de MongoDB et Redis révèle des différences de performances attribuables à leurs architectures distinctes. Redis, avec son stockage en mémoire, montre une supériorité en termes de rapidité, offrant des lectures et des mises à jour environ 50% plus rapides que MongoDB (selon les données des tableaux), ce qui confirme son efficacité en tant que solution de stockage clé-valeur pour les applications nécessitant un accès immédiat aux données. Ceci est renforcé par son modèle *single-threaded* qui, malgré les craintes de goulot d'étranglement, se révèle efficace grâce au multiplexage I/O pour gérer de nombreuses connexions.

D'autre part, MongoDB brille lorsqu'il s'agit de gérer des montées en charge de lectures, indiquant une capacité accrue à traiter des requêtes complexes et des volumes importants de données, grâce à son format BSON.

Cependant, des limitations ont été observées dans notre travail, notamment la performance sous charges de travail lourdes et la scalabilité dans les configurations de clusters étendus.

Les principales difficultés rencontrées lors de l'expérimentation ont été la configuration et les particularités de l'outil YCSB. Par exemple, la version la plus récente de MongoDB n'était pas compatible avec YCSB, mais ce n'était pas nécessairement évident initialement avec les messages d'erreurs qui étaient retournés. Une fois des recherches effectuées et la source du problème trouvé, nous avons pu changer la version de MongoDB dans le fichier Docker Compose afin de régler le problème.

Il existe certains obstacles à la validité de nos données, en particulier les données aberrantes et à quel point l'environnement de test est représentatif des conditions réelles d'utilisation. Les données abérrantes sont possiblement liés à de l'instabilité dans l'environnement de test et elles agrandissent nos intervalles de confiance, ce qui limite notre capacité à faire des observations avec certitude. Cependant, dans la grande majorité des cas, les intervalles sont tout de même assez petits pour pouvoir distinguer clairement une différence de performance entre MongoDB et Redis.

Malgré ces défis, les résultats nécessitent une analyse critique, compte tenu des limitations inhérentes aux environnements de test simulés. Les performances réelles peuvent varier en fonction de nombreux facteurs non capturés dans un environnement de laboratoire, tels que les conditions réseau,

la charge de travail simultanée, et la configuration matérielle spécifique.

En synthèse, tout en reconnaissant la performance impressionnante de Redis en termes de rapidité, MongoDB se distingue par sa capacité à traiter de grandes charges de données avec complexité. La décision entre les deux doit être guidée par une analyse précise des besoins fonctionnels, en tenant compte des différences de performances observées et des exigences opérationnelles.

V. Conclusion

Cette étude comparative entre MongoDB et Redis a mis en évidence des différences notables en termes de performances, de structures de données, et de gestion des opérations. Redis, avec son modèle de stockage en mémoire et ses structures de données optimisées, excelle dans les opérations de lecture et d'écriture rapide, se prêtant idéalement à des applications exigeant un accès immédiat aux données, telles que le caching ou la gestion de sessions. D'un autre côté, MongoDB, avec son stockage majoritairement sur disque et sa flexibilité offerte par le format BSON, se révèle être mieux adapté pour les applications nécessitant des requêtes complexes et la gestion de grands volumes de données avec une intégrité accrue.

Nos expérimentations ont clairement démontré que chaque système possède des avantages spécifiques qui doivent être pris en compte selon les besoins particuliers de chaque application. MongoDB se distingue dans les scénarios qui impliquent des transactions complexes et une persistance des données, tandis que Redis offre une rapidité et une efficacité énergétique supérieures pour des tâches plus simples et immédiates.

À l'avenir, il serait pertinent d'explorer des architectures hybrides combinant les forces de MongoDB et Redis pour maximiser la performance et l'efficacité des systèmes de gestion de bases de données dans divers contextes d'application. Cette étude enrichit notre compréhension des systèmes de gestion de bases de données NoSQL et offre une base solide pour des décisions technologiques éclairées, face aux défis contemporains de stockages et de manipulations de données à grande échelle.

References

[1] Trujillo, G., Kim, C., Jones, S., Garcia, R.et Murray, J. (2015). *Virtualizing hadoop: how to install, deploy, and optimize hadoop in a virtualized architecture*. VMware Press

[2] Ji, C., Li, Y., Qiu, W., Jin, Y., Xu, Y., Awada, U., ... et Qu, W. (2012). *Big data processing: Big challenges and opportunities. Journal of Interconnection Networks*, 13(03n04), 1250009. https://doi.org/10.1142/S0219265912500090

[3] Zobaed, S.M., Salehi, M.A. (2022). *Big Data in the Cloud*. Springer, Cham. https://doi.org/10.1007/978-3-319-63962-8_40-2

[4] Matallah, H., Belalem, G., et Bouamrane, K. (2017). *Experimental comparative study of NoSQL databases: HBASE versus MongoDB by YCSB*. Comput. Syst. Sci. Eng, 32(4), 307-317.

[5] Khazaei, H., Fokaefs, M., Zareian, S., Beigi-Mohammadi, N., Ramprasad, B., Shtern, M., ... et Litoiu, M. (2016). *How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey*. Big Data & Information Analytics.https://doi.org/10.3934/bdia.2016004

[6] Dipina Damodaran, B., Salim, S., et Vargese, S. M. (2016). *Performance evaluation of MySQL and MongoDB databases*. Int. J. Cybern. Inform.(IJCI), 5, 387-394.https://www.aircconline.com/ijci/V5N2/5216ijci41.pdf

[7] Chowdary, R. (2023). *MongoDB: Say No to SQL*. Medium. https://ravisrc.medium.com/mongodb-say-no-to-sql-8ef7b94a6ae9

[8] Paksula, M. (2010). *Persisting objects in redis key-value database*. University of Helsinki, Department of Computer Science. https://www.cs.helsinki.fi/u/paksula/misc/redis.pdf

[9] Borisov, B.(2021) *Redis as Cache: How it Works and Why to Use it*.https://linuxiac.com/redis-as-cache/

[10] Amazon. (s.d.).*What's the Difference Between Redis and MongoDB?*. https://aws.amazon.com/compare/the-difference-between-redis-and-mongodb/

[11] MongoDB. (s.d.) *MongoDB vs. Redis Comparison*. https://www.mongodb.com/compare/mongodb-vs-redis

[12] Redis. (s.d.) Redis. https://redis.io

[13] Seghier, B., Kazar, O. (2021) *Performance Benchmarking and Comparison of NoSQL Databases: Redis vs MongoDB vs Cassandra Using YCSB Tool*, 2021 International Conference on Recent Advances in Mathematics and Informatics (ICRAMI), Tebessa, Algeria, 2021, pp. 1-6, https://ieeexplore.ieee.org/document/9585956.

[14] Patel, R. (2021). Data+ Education. Redis Is a Cache or More?. EasyChair Preprint, 88.

[15] Rasolroveicy, M., Fokaefs, M. (2020). *Performance Evaluation of Distributed Ledger Technologies for IoT data registry : A Comparative Study* 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 2020, pp. 137-144. https://ieeexplore.ieee.org/abstract/document/9210358

[16] Araujo, J. M. A., de Moura, A. C. E., da Silva, S. L. B., Holanda, M., de Oliveira Ribeiro, E., et da Silva, G. L. (2021). *Comparative performance analysis of NoSQL Cassandra and MongoDB databases*. 2021 16th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). https://ieeexplore.ieee.org/abstract/document/9476319

[17] Mansouri, Y., Prokhorenko, V., Ullah, F., et Babar, M. A. (2023). *Resource utilization of distributed databases in edge-cloud environment*. IEEE Internet of Things Journal.https://ieeexplore.ieee.org/abstract/document/10012453