Comparaison entre NoSQL et Blockchain en tant que bases de données distribuées

Adam Halim
dept. Informatique et Logiciel
Polytechnique Montréal
Montréal, Canada
adam.halim@polymtl.ca

Oussama Lourhmati
dept. Informatique et Logiciel
Polytechnique Montréal
Montréal, Canada
oussama.lourhmati@polymtl.ca

Liêm Truong
dept. Informatique et Logiciel
Polytechnique Montréal
Montréal, Canada
liem.truong@polymtl.ca

Hee-Min Yang
dept. Informatique et Logiciel
Polytechnique Montréal
Montréal, Canada
hee-min.yang@polymtl.ca

Abstrait— Aujourd'hui, la question de savoir s'il est possible de stocker l'information ne se pose plus. En effet, les solutions à ce problème sont abondantes : bases de données locales, stockées sur le cloud, relationnelles, NoSQL et plus encore. Bref, les choix ne manquent pas. L'enjeu actuel concerne plutôt la performance des bases de données. Justement, les bases de données NoSQL offrent une hausse de performance significative par rapport aux bases de données relationnelles. Effectivement, en évitant le stockage des données sous format tabulaire, celles-ci adoptent des structures flexibles, accélérant la lecture et l'écriture des données [7]. Plus récemment, l'avènement de la cryptomonnaie a inspiré le développement des bases de données blockchain. En se basant sur le même principe de distribution des données parmi les utilisateurs, les bases de données blockchain pourraient également bénéficier d'une performance compétitive. Le présent article a pour but de comparer la performance des bases de données NoSQL et blockchain.

Mots-clés— Base de données, NoSQL, Blockchain, Hyperledger, MongoDB, Performance, Benchmark

I. Introduction

Ces dernières années, avec l'arrivée de l'Internet des objets (IoT) et de l'infonuagique (cloud computing), le besoin d'espaces de stockage pour stocker de manière efficace et rentable de grandes quantités de données a considérablement augmenté. Cet ensemble de grande quantité de données correspond au Big Data, et peut être parfois assez complexe. En effet, les applications de traitement traditionnelles ne peuvent pas traiter autant de données. Lorsque nous parlons de données volumineuses, nous pouvons évoquer les quatre V qui sont le volume, la vélocité, la variété ainsi que la véracité. Le volume représente le grand volume de données et de sources. La vélocité des données correspond à la vitesse auxquelles celles-ci sont générées et traitées. La variété fait référence au type et à la nature des données. La véracité s'agit de la qualité des données stockées pouvant significativement affecter la précision de l'analyse [1].

De ce fait, il existe plusieurs architectures logicielles pour le Big Data. Le système blockchain peut être perçu comme une base de données distribuée. Une autre option serait simplement une base de données NoSQL standard pouvant s'avérer tout aussi utile.

A. Motivation

La principale motivation derrière l'utilisation de bases de données distribuées est la gestion de grands volumes de données tout en garantissant une disponibilité en tout temps. Pour cette raison, il est essentiel que le système soit réparti sur plusieurs serveurs et que les données soient partitionnées et partagées [2]. En fait, les bases de données distribuées peuvent être définies en tant qu'une collection de plusieurs bases de données interdépendantes et réparties sur un réseau. Ces dernières offrent une multitude d'avantages dont l'évolutivité. En effet, elles peuvent être mises à l'échelle horizontalement en ajoutant des nœuds au réseau afin d'augmenter la capacité et la performance à mesure que le nombre de données augmentent. Elles offrent aussi une disponibilité accrue en répartissant les données sur plusieurs nœuds. Ainsi, si un nœud tombe en panne, les données seront toujours accessibles à partir d'autres nœuds répliqués. Il y a donc aussi une tolérance aux pannes puisqu'il existe des mécanismes de redondance et de basculement permettant au système de continuer à fonctionner si un nœud est défaillant. La grande flexibilité permet aussi le stockage de données de la manière la plus adaptée selon l'application ou les utilisateurs. Finalement, les bases de données distribuées sont plus sécurisées que les bases de données centralisées sachant que la distribution et les transactions des données sont transparentes aux utilisateurs. Bref, toutes ces raisons peuvent pousser les utilisateurs à opter pour un tel système de base de données [3].

B. Système blockchain comme base de données distribuée

Le blockchain peut agir comme un système de base de données distribué et décentralisé. L'immuabilité, la transparence, la confidentialité et la sécurité sont des caractéristiques d'un tel système. La connexion entre les nœuds au sein d'un réseau se fait en P2P (peer-to-peer) de manière décentralisée et sans l'interception d'un administrateur. Il y a donc un consensus à atteindre entre les utilisateurs. En fait, il s'agit d'un algorithme où tous les nœuds atteignent un accord commun sur l'état du système. Les données sont sauvegardées en blocs en utilisant une signature cryptographique sécurisée. Cependant, ces données ne peuvent pas être directement modifiées. En effet, il est seulement possible de mettre à jour l'état du système tout en préservant une trace de toutes les transactions dans un livre numérique distribué et immuable. De plus, ces données peuvent être partagées mais sont accessibles à tous selon des autorisations bien précises [4].

Les avantages d'un tel système comparé à une base de données traditionnelle sont nombreux. Tout d'abord, il y a un gain de temps pour les virements et transactions puisqu'il n'y a pas d'intermédiaire central entre les nœuds. De ce fait, il y a une réduction des coûts. De plus, il y a aussi une garantie de la confidentialité, de l'intégrité et de la disponibilité du système. Finalement, tel que mentionné, le Blockchain offre une traçabilité améliorée et une transparence au niveau des données. Ces dernières restent visibles pour les utilisateurs impliqués dans le réseau [4].

C. Base de données non relationnelle (NoSQL)

Les bases de données non relationnelles ou encore appelées NoSQL sont différentes des bases de données traditionnelles. En effet, elles stockent leurs données sous une forme non tabulaire. Au lieu d'avoir des tables avec des lignes et colonnes, ces dernières sont basées sur des structures de données telles que des documents. Un document peut s'avérer très complexe et détaillé contenant souvent une multitude d'informations dans plusieurs formats. Cette capacité à organiser autant de types d'informations rend ce type de base de données beaucoup plus flexible et adaptable qu'une base de données relationnelle. Par ailleurs, il existe plusieurs types de bases de données NoSQL tels qu'un document, des paires clé-valeur ou sous forme de graphique.

L'utilisation de ce genre de bases de données comporte plusieurs avantages notamment l'organisation d'ensembles de données massives. Les bases de données non relationnelles peuvent non seulement stocker de très grandes quantités d'informations mais gèrent aussi les requêtes avec facilité. La mise à l'échelle et la vitesse de performance sont donc des avantages significatifs. Il faut aussi se rappeler que les données ne sont pas statiques. En effet, au fur et à mesure que la collecte d'informations s'élargit, une base de données NoSQL peut facilement enrichir la base de données existante, et ce, même avec des types de données différentes. De plus, elles prennent en charge toutes structures de données possibles, que ce soit des caractères, des chiffres ou des vidéos et photos. Par ailleurs, les bases de données NoSQL sont conçues pour l'infonuagique et permettent l'évolutivité au cloud de manière exponentielle [5].

D. Objectifs et contributions

Le principal objectif de cet article est de savoir comment un système blockchain en tant que base de données distribuée se compare à une base de données NoSQL standard en termes de performances. Il est important d'indiquer que nous ignorons l'aspect de sécurité dans cette comparaison détaillée. Il sera aussi question de mieux comprendre et d'analyser quels facteurs pourraient influencer les différences de performances s'il y en a. Ainsi, il sera plus facile de conclure quel serait le type de base de données approprié selon le contexte et les besoins de l'utilisateur. De plus, les performances seront mesurées à l'aide d'outils bien adaptés et selon plusieurs critères tels que la latence de lecture/écriture, le débit de lecture/transaction et les métriques de consommations de ressources (CPU, mémoire, réseau).

II. Contexte

Le survol des technologies précédentes permet d'apprécier la diversité des solutions disponibles quant aux bases de données distribuées. Tel qu'il a été expliqué précédemment, l'article vise à comparer la performance entre un système blockchain comme base de données et une base de données NoSQL. Hyperledger Fabric a été retenu pour le premier, et pour le dernier, MongoDB a été choisi.

A. Hyperledger Fabric

Le système blockchain utilisé comme base de données dont il sera question dans cet article est Hyperledger Fabric. Ce dernier est en fait un ensemble d'outils et de librairies pouvant être utilisés pour créer une application de type blockchain pour répondre à des besoins précis d'une entreprise [10]. Il s'agit d'une implémentation particulière du blockchain, partageant des similarités avec des technologies comme la cryptomonnaie, mais ayant ses propres caractéristiques qui le distingue des systèmes de registres distribués couramment connus. Hyperledger Fabric est un projet à source ouverte de la Fondation Linux bénéficiant de la contribution de compagnies de renommée internationale comme IBM. Il s'agit d'une solution largement utilisée en industrie avec des applications commerciales [8].

La plupart des systèmes blockchain mis en place ont été conçus pour usage public, c'est-à-dire sans gouvernance ni confidentialité [9]. Bien que cela permette d'avoir une transparence complète [10], cette dernière n'est pas toujours désirable. En effet, si un client fait affaire avec plusieurs fournisseurs, il pourrait ne pas vouloir que ses transactions avec un fournisseur soient visibles aux autres fournisseurs. En réponse à ce cas d'utilisation, Hyperledger Fabric offre la possibilité de créer des sous-réseaux. Dans l'exemple précédent, le client ferait partie d'un réseau avec l'ensemble

des fournisseurs, et pourrait en plus communiquer avec des fournisseurs particuliers via des canaux spécifiques, permettant ainsi d'isoler les transactions. Pour augmenter la confidentialité, Hyperledger Fabric permet en plus de transmettre des données hashées lors de demandes d'accès spécifiques seulement [9].

Un réseau Fabric est composé d'organisations uniques, chacune représentant un ensemble d'utilisateurs. Une organisation a au moins un nœud pair auquel sont délégués les opérations qui lui sont liées. En particulier, le nœud pair approuve les transactions d'un utilisateur, stocke et exécute des contrats de manière automatique et garde une copie du registre de transaction pour accès par d'autres clients. Les organisations possèdent des certificats racines qui leur sont propres, et les utilisateurs des organisations ont des certificats qui dérivent de ces certificats racines, permettant de les authentifier et de les identifier auprès d'autres organisations. Les certificats définissent les permissions du détenteur, par exemple pour les permissions de lecture et d'écriture. C'est l'autorité de certificat du réseau Fabric qui octroie les certificats aux organisations et leurs membres, et qui gère tout autre opération liée aux certificats. Enfin, le réseau Fabric contient un service d'ordonnancement qui se charge du séquencement des transactions et de vérifier qu'elles soient validées par l'organisation correspondante [11]. Le schéma suivant illustre l'interaction entre les différentes entités dans le cas d'un réseau pour une application mobile de soins de santé.

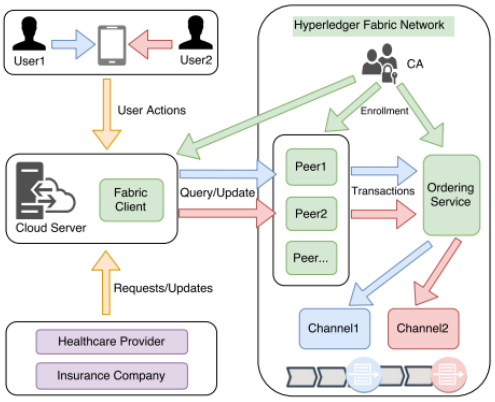

Fig. 1. Interactions entre entités d'un réseau Fabric [12]

Les composantes d'un réseau Fabric étant définies, il est maintenant possible de résumer les étapes suivies lors de la soumission d'une transaction. Le client commence par soumettre une proposition de transaction aux nœuds pairs des organisations. Ces pairs vérifient d'abord l'identité et l'autorité derrière le client, puis simulent la transaction pour s'assurer de leur validité. Si la transaction est valide, le pair renvoie une signature d'approbation au client. Lorsque le client reçoit assez de signatures, il envoie la transaction au service d'ordonnancement. Ce dernier vérifie que la transaction ait le nombre nécessaire d'approbations, puis ordonne les transactions en blocs qui sont alors envoyés aux nœuds pairs. Finalement, ces nœuds effectuent une dernière validation puis ajoutent les blocs au registre [11]. Les étapes sont illustrées dans la figure ci-dessous.

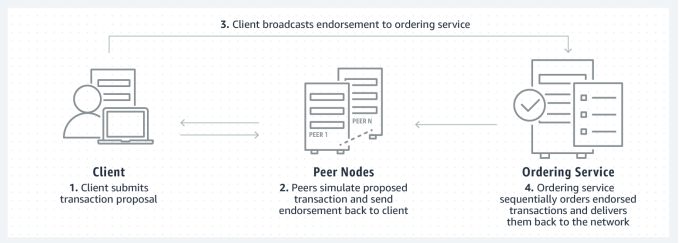

Fig. 2. Étapes de transaction dans Hyperledger Fabric [11]

L'architecture de Hyperledger Fabric permet de déduire ses caractéristiques principales. Son fonctionnement est basé sur les permissions, c'est-à-dire que toutes les entités impliquées dans un réseau sont authentifiées et leur identité connue. De plus, le contrôle d'accès est régulé dans les canaux privés et non par une autorité centrale. Ces caractéristiques permettent de garder la confidentialité des transactions tout en gardant leur traçabilité. Finalement, le consensus basé sur l'attribution de signatures par les pairs offre une meilleure performance que les méthodes traditionnelles de consensus comme la preuve de travail ou la preuve d'enjeu [11].

B. MongoDB

La base de données non relationnelle qui sera évaluée est MongoDB. Ce qui distingue MongoDB des autres bases de données non relationnelles est qu'elle stocke l'information dans des documents. Un document représente généralement un objet et ses métadonnées, tel qu'un utilisateur avec son adresse courriel, numéro de téléphone, et ses passe-temps favoris. Ainsi, un document est une structure de données avec des paires de clés-valeurs, et peut être formaté en BSON, en XML ou en JSON qui est désormais le format standard. Un ensemble de documents peut être regroupé dans une collection. Souvent, les documents seront liés sémantiquement, mais ne doivent pas nécessairement contenir les mêmes champs, bien qu'un schéma définissant les champs peut être imposé pour uniformiser les documents d'une collection [13].

La facilité d'utilisation de MongoDB est une de ses caractéristiques importantes. En effet, puisque les documents sont formatés de manière intuitive pour le développeur (en format JSON en l'occurrence), il est facile de les lier aux objets dans le code. De plus, MongoDB implémente des API permettant d'effectuer des opérations de type CRUD (*create*, *read*, *update*, *delete*) [13] ce qui facilite davantage le travail des développeurs. Par ailleurs, MongoDB permet la mise à l'échelle horizontale grâce au principe de *sharding* intégré automatiquement aux bases de données. Ce principe consiste à distribuer les données à travers des partitions physiques (appelées *shards*) en fonction de la taille des données et de la grappe de *shards*

partitionnée [14]. La figure ci-dessous montre la mise à l'échelle horizontale qui peut se faire grâce au *sharding*.

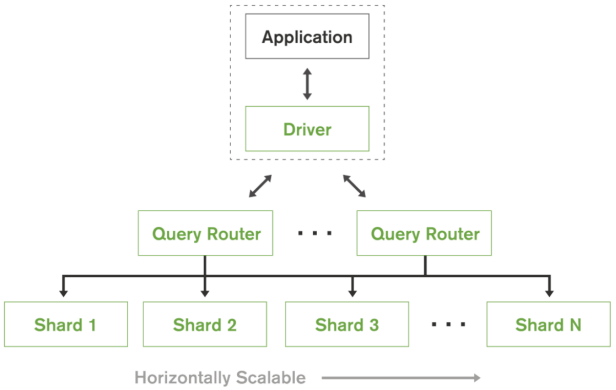

Fig. 3. Mise à l'échelle horizontale dans MongoDB [14]

Un *shard* est un ensemble de copies de données appelé *replica set*. L'ensemble de répliques est divisé en une réplique primaire et des répliques secondaires. Par défaut, la réplique primaire reçoit et traite toutes les demandes de lecture et d'écriture. Si la réplique primaire tombe en panne, les requêtes sont basculées vers une réplique secondaire qui est élue comme primaire, assurant ainsi la disponibilité des données. Ce processus est automatique, caractérisant les répliques d'auto-guérissantes (*self-healing*), comme l'illustre la figure suivante [14].

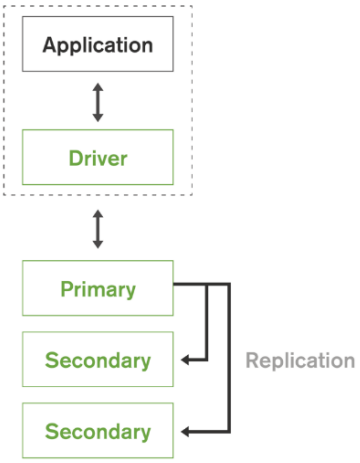

Fig. 4. Répliques auto-guérissantes dans MongoDB [14]

Finalement, MongoDB inclut plusieurs types de moteurs de stockage pour répondre à des besoins de natures différentes. En particulier, le moteur *In-Memory* permet de d'obtenir la performance équivalente au chargement des données en mémoire vive avant de stocker les changements dans la mémoire disque persistante. Cela diminue considérablement la latence et augmente la performance de la base de données, tout offrant des fonctionnalités supplémentaires comme l'analyse de métriques en temps réel [14].

C. Comparaison

Bien qu'ils soient tous deux des bases de données distribuées, Hyperledger Fabric et MongoDB diffèrent largement en termes d'architecture, de propriétés de base et de fonctionnalités. À partir des descriptions présentées précédemment, il est possible de souligner les forces et faiblesses des deux technologies dans le tableau suivant.

TABLEAU I. COMPARAISON ENTRE HYPERLEDGER FABRIC ET MONGODB

Base de donnée	***Forces***	***Faiblesses***
Hyperledger Fabric	- Assure la confidentialité entre les parties impliquées - Système de consensus garantit l'authenticité et la traçabilité et améliore la performance comparativement au blockchain traditionnel	- Plus difficile à implémenter et à comprendre que MongoDB - Solution surtout appropriée pour les moyennes-grandes entreprises
MongoDB	- Facile et rapide à implémenter - Système de *sharding* efficace pour la redondance et la disponibilité - Diversité des moteurs de stockage donne la flexibilité et la performance	- Supporte seulement les données sous forme de documents - Le manque de schéma obligatoire peut mener à une hétérogénéité des documents augmentant le risque d'erreurs

III. MÉTHODOLOGIE

La méthodologie de travail consiste à exécuter un total de 3 tests différents pour chaque combinaison de charge de travail (uniquement en lecture, 50/50 en lecture/écriture, 10/90 lecture/écriture), et ce, sur chacune des bases de données à l'aide des outils de benchmarking. Il sera plus facile d'établir une moyenne des résultats pour chaque base de données afin de garantir des valeurs finales plus précises, fiables et représentatives. L'outil Hyperledger Caliper permet d'évaluer les performances de la base de données Hyperledger Fabric en ce qui concerne la latence, le débit et la consommation de ressources. Tandis que l'outil Yahoo! Cloud Service Benchmark (YCSB), qui est libre d'accès, sera utilisé pour comparer les performances relatives de la base de données NoSQL. Il sera aussi essentiel d'installer les bibliothèques requises et de configurer l'environnement virtuel afin d'exécuter les tests. Une fois l'exécution terminée, il sera plus facile d'établir une comparaison complète entre les deux types de bases de données.

IV. EXPÉRIMENTATION

Afin d'assurer des environnements de tests fonctionnels et isolés pour les bases de données (Hyperledger Fabric et MongoDB), nous avons déployé deux instances de serveur EC2 de AWS différentes. Ceci permettra de limiter les variations et les erreurs dues à des réglages de configuration d'un système pouvant avoir un impact sur le fonctionnement

d'un autre système. En effet, les deux types de bases de données ne sont pas configurées et déployées de la même manière, et n'utilisent pas les mêmes versions des librairies. Il est donc plus judicieux de les isoler dans leurs propres environnements. De ce fait, une distribution Ubuntu 20 a été utilisée pour MongoDB et YCSB, puisqu'il s'agit de la version compatible avec ces dernières. Pour le Hyperledger, la distribution Ubuntu 22 a été déployée.

A. Configuration

Pour chacune des bases de données testées, nous avons défini un nombre précis de nœuds. Pour MongoDB, nous avons un total de 4 nœuds, dont 1 qui est primaire avec 3 réplicas. Du côté de Hyperledger, nous avons 2 organisations (org1 et org2) avec un peer par organisation. L'expérimentation a été exécutée sur des instances AWS de type T2.large (2 vCPU et 8 Go de RAM) avec 20 Go de stockage, et cela, sous Ubuntu, tel que mentionné précédemment. En effet, un minimum de 8 Go de RAM est nécessaire pour s'assurer que les tests se déroulent sans problème. Afin de pouvoir répliquer l'étude, il suffit simplement de suivre les étapes de configuration dans les fichiers README propres aux différentes bases de données.

B. Déploiement et infrastructure

Un dépôt GitHub est utilisé pour stocker les fichiers de configurations et les étapes nécessaires [6]. Tel qu'indiqué dans ces étapes, il sera essentiel d'installer toutes les dépendances requises avant de pouvoir effectuer les tests. En effet, ces derniers nécessitent, entre autres, l'installation de Python 2.7, de Git pour cloner les dépôts, de Docker et Docker Compose pour les bases de données conteneurisées, de paquets build-essential et plus encore. Les sorties pour chaque test effectué de chaque base de données ont été sauvegardées sous forme de fichiers dans le dépôt.

Pour MongoDB, une fois la configuration terminée à l'aide du docker-compose-mongodb.yaml présent dans le dépot GitHub, 3 tests différents seront exécutés pour chaque combinaison de charge de travail pour un total de 9 tests, afin de calculer une moyenne des valeurs de sortie avec précision. Pour chaque nœud, un conteneur Docker sera créé et les tests seront lancés par la suite. De plus, tel que mentionné, un seul nœud primaire sera actif avec un total de 3 secondaires. Le port ouvert pour chacun est le 27017. Les adresses IP utilisées pour chaque nœud sont toutes dans le même sous réseau que l'adresse IP de l'instance EC2 de AWS.

Pour Hyperledger Fabric, plusieurs fichiers de configuration YAML sont déjà présents par défaut dans le dépôt Git qui est cloné dans les étapes de configuration. En effet, il suffit d'exécuter la liste des commandes présentes dans le README, étape par étape, pour obtenir un environnement de test fonctionnel [6]. Par ailleurs, deux organisations (org1 et org2), contenant chacune un peer, seront actives. Chacune de ces organisations agit comme une entité ayant accès aux canaux (channels) tout en vérifiant les identités et la source de chaque transaction. Les canaux sont essentiellement des registres utilisés pour stocker et sécuriser les données.

C. Tâches et workloads

Afin d'analyser avec justesse la performance des bases de données, les mêmes tâches ont été exécutées pour les deux bases de données avec les mêmes workloads. En particulier, 3 workloads différents sont définis pour les tests. Chacun consiste à faire des opérations de lecture (READ) et d'écriture (WRITE) en faisant varier les proportions afin de simuler des cas d'utilisation différents. Pour des fins de comparaison, il est important de noter qu'une écriture dans MongoDB correspond à une transaction blockchain. Les workloads sont décrits dans le tableau suivant.

TABLEAU II. WORKLOADS UTILISÉS POUR LES TESTS

Workload	*Operation*	*Pourcentage*	*Caractéristique*
Workload A	READ	50%	Charge équilibrée
	WRITE	50%	
Workload B	READ	10%	Écriture majoritaire
	WRITE	90%	
Workload C	READ	100%	Lecture seulement
	WRITE	0%	

Chaque workload consiste à effectuer 1000 opérations, correspondant au nombre d'opérations par défaut avec l'outil YCSB [15]. Ainsi, le workload A comporte 500 lectures et 500 écritures, le workload B comporte 100 lectures et 900 écritures et le workload C comporte 1000 lectures. Notons que l'exécution des workloads est toujours précédée par un chargement des données. L'analyse de la performance ne prendra pas cette étape en compte et se limitera seulement à l'exécution des opérations définies dans les workloads.

D. Configuration des outils de test

L'exécution des tests pour MongoDB se fait avec YCSB. Chaque test est exécuté en deux commandes; une pour effectuer le chargement des données (load) et une pour exécuter le workload (run). Pour chaque combinaison de charge de travail, il existe déjà des fichiers de configuration YCSB sous les noms de workloada (50/50 en lecture/écriture), workloadb (10/90 lecture/écriture) et workloadc (uniquement en lecture).

Hyperledger Caliper est utilisé pour effectuer les tests de Hyperledger Fabric. Le fichier config.yaml présent dans le dépôt peut être utilisé pour configurer les différentes combinaisons de charge de travail, en modifiant les paramètres txDuration, txNumber et tps. Ensuite, une fois le fichier configuré, il suffit simplement de retourner à la

source de projet et de rouler la dernière commande pour exécuter le test.

E. Resultats

Pour chaque base de données, nous avons analysé la latence moyenne des opérations de lecture et d'écriture. Le débit a également été analysé. 1000 opérations sont exécutées par charge de travail. Le graphique ci-dessous présente les résultats de nos analyses. Notons que la latence est mesurée en microsecondes.

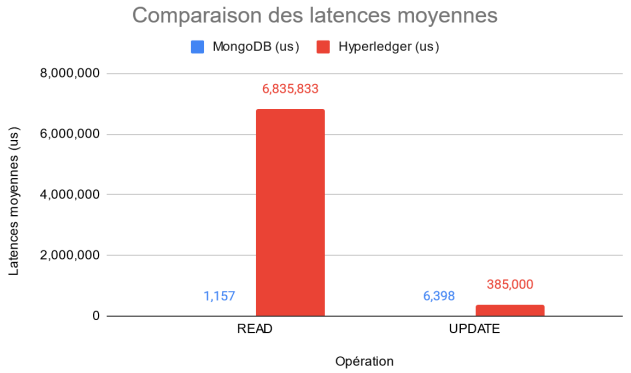

Fig. 5. Comparaison des latences moyennes des bases de données pour les opérations READ et UPDATE

TABLEAU III. LATENCE MOYENNE PAR OPÉRATION EN MICROSECONDES

BD /Opération	*MongoDB (us)*	*Hyperledger (us)*
READ	1,157	6,835,833
UPDATE	6,398	385,000

En analysant les différentes métriques, graphiques et tableaux, il est possible d'y extraire quelques observations par rapport à leur performance relative.

En se fixant d'abord sur la figure 5 et le tableau III, nous voyons que le temps de latence Hyperledger peu importe l'opération est très long (ordre de grandeur de 10^3 à 10^6). Spécifiquement, pour l'opération *READ*, Hyperledger a un temps de latence de 6835833 us tandis que celle de MongoDB est de 1157 us. Pour l'opération *UPDATE*, le temps de latence de Hyperledger est de 385000 et celui de MongoDB est de 6398.

Au lieu de comparer les latences selon les différentes opérations, il est aussi possible de les comparer selon la répartition de charge de travail. Considérant que la charge de travail A est de 50 *READ*/50 *UPDATE*, B 10 *READ*/90 *UPDATE* et C 100 *READ*/0 *UPDATE*, le tableau IV ainsi que la figure 6 et le tableau IV confirment que MongoDB a toujours une latence moins élevée (plus performante) peu importe la répartition de la charge de travail. Toutefois, il est important de noter que la latence de la charge de travail A de Hyperledger est significativement plus élevée que les autres. Bien que cette donnée pourrait être une erreur, même en l'omettant, MongoDB serait plus performant que Hyperledger.

TABLEAU IV. LATENCE MOYENNE PAR CHARGE DE TRAVAIL EN MICROSECONDES

BD/Charge de travail	*MongoDB (us)*	*Hyperledger (us)*
A	4,800	20,040,000
B	4,373	417,500
C	3,252	50,000

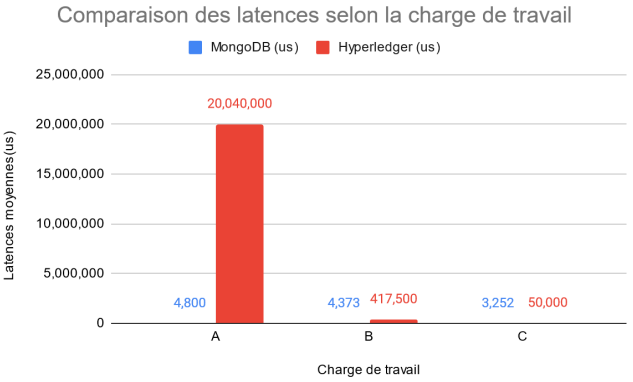

Fig 6. Comparaison des latences moyennes des bases de données selon différentes répartitions de charge de travail

La latence représente le temps requis pour le traitement d'une requête par la base de données. On considère qu'une latence relativement élevée signifie que la base de données est lente, puisque le temps nécessaire pour traiter une requête est également élevé.

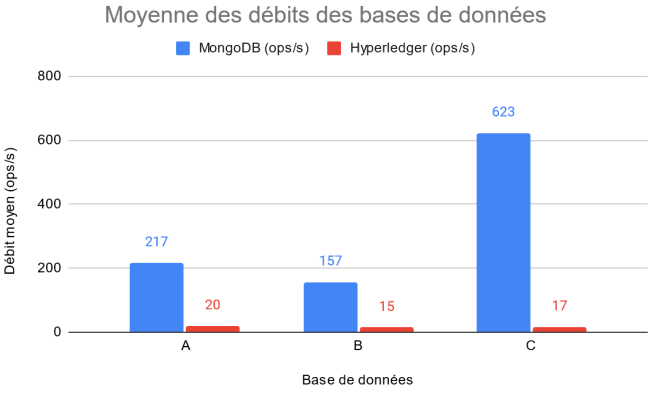

Fig. 7. Comparaison des débits des bases de données selon différentes répartitions de charge de travail

Par la suite, à l'aide de la figure 7 et du tableau V, il est possible de comparer le débit total selon les différentes distributions de charge de travail. Plus le débit est haut, plus le nombre d'opérations est possible signifiant une performance plus élevée. Considérant que la charge de travail A est de 50 *READ*/50 *UPDATE*, B 10 *READ*/90 *UPDATE* et C 100 *READ*/0 *UPDATE*, MongoDB a toujours un débit plus élevé que Hyperledger (ops/s). Effectivement, pour les charges de travail A,B et C, MongoDB a des débits de 217 ops/s, 157 ops/s et 623 ops/s respectivement tandis

que Hyperledger a des débits de 20 ops/s, 15 ops/s et 17 ops/s. Les débits de MongoDB sont environ 100 à 300 fois plus performants que ceux de Hyperledger. MongoDB a alors le débit le plus performant.

À la lumière de nos résultats, nous pouvons constater que MongoDB est la base de données la plus performante. En effet, la latence moyenne des opérations est beaucoup plus petite que celle de Hyperledger et ce, d'un facteur d'environ 6000 pour la lecture et 60 pour l'écriture.

TABLEAU V. DÉBITS PAR CHARGE DE TRAVAIL EN OPS/S

BD/Charge de travail	*MongoDB (ops/s)*	*Hyperledger (ops/s)*
A	217	20
B	157	15
C	623	17

V. DISCUSSION

En se référant à la figure 5, il est évident que le temps de latence pour l'opération *READ* pour la base de données Hyperledger est très grand. Effectivement, lors de nos tests, le temps de latence maximal pour la charge de travail A était de 24.06 s tandis qu'il était de 0.47 pour la charge de travail B et environ 0.11 pour la charge de travail C. Il est alors possible que les résultats obtenus pour la charge de travail A soient anormalement élevés et ont alors faussé nos résultats. Toutefois, même si on réduisait notre latence maximale à un temps plus réaliste de 5 secondes, MongoDB serait tout de même plus performant à un ordre de grandeur considérable.

Si on ignore le fait que cette donnée est possiblement aberrante, nous voyons que les résultats pour les débit par charge de travail concordent avec ceux des latences. Effectivement, la latence est inversement proportionnelle au débit des opérations. Plus le temps d'exécution d'une opération est court, plus on peut faire d'opérations en un intervalle de temps. En analysant les figures 6 et 7, on constate que MongoDB a des latences plus courtes et des débits plus élevés comparé à Hyperledger.

Par contre, sachant que la latence est inversionnellement proportionnelle au débit des opérations, une autre incohérence est présente. Effectivement, la latence de Hyperledger pour la charge de travail A est significativement plus longue comparée aux autres, mais le débit pour la charge de travail A est aussi le plus élevé. Habituellement, le débit de la charge de travail A devrait être le plus faible.

En observant les différents graphiques et tableaux, MongoDB est clairement le plus performant des deux bases de données. Ceci s'explique par le fait que MongoDB est une base de données basée sur les documents, aussi connue comme une base de données *NoSQL* [16]. Effectivement, lorsqu'on veut modifier ou lire une donnée spécifique, il n'y a aucun besoin de chercher à travers plusieurs tables comme le fait une base de données SQL. La base de données ne fait que lire le contenu d'un document qui contient habituellement tous les champs nécessaires.

Comme mentionné précédemment, Hyperledger étant une base de données de type système blockchain doit suivre plusieurs étapes à chaque transaction. Effectivement, comme le démontre la figure 2, lors d'une transaction, les entités impliquées doivent approuver la transaction et doivent par la suite mettre en ordre les transactions. Ces étapes garantissent la sécurité en échange d'un temps de transaction plus long [11]. Nos résultats démontrant que MongoDB est plus performant en latence et débit des opérations coïncident alors avec la documentation; les bases de données NoSQL sont optimisés pour les transactions rapides avec leurs documents contenant plusieurs champs tandis que les systèmes blockchain sont moins rapides puisqu'ils sont optimisés pour la sécurité, intégrité et confidentialité des données.

Quant aux limites de nos méthodes, il faut noter que les manipulations n'ont été faites que sur deux implémentations de bases de données, soit une base de données NoSQL et un système blockchain. Il est possible que les implémentations spécifiques des bases de données (MongoDB et Hyperledger) puissent présenter des particularités comparées aux autres bases de données. Il se peut alors que nos résultats permettent une comparaison des implémentations spécifiques plutôt que d'une comparaison générale des bases de données NoSQL et d'un système blockchain. De plus, le nombre d'opérations dans nos tests était limité à 1000. En industrie, le nombre d'opérations dépasserait facilement cette limite. Ceci étant dit, il se peut que notre échantillon soit considéré trop petit pour que nos résultats soient considérés pertinents.

VI. CONCLUSION

En conclusion, nous avons fait l'analyse de deux différents types de base de données et avons présenté diverses caractéristiques sur celles-ci. Nous avons comparé Hyperledger Fabric, un système blockchain comme base de données, à MongoDB, une base de données NoSQL. Un des défis du Big Data est de stocker de grandes quantités de données en ayant de bonnes performances. Pour cette raison, nous avons effectué plusieurs tests de performance sur les deux bases de données mentionnées dans des environnements contrôlés. Les bases de données ont été déployées dans deux instances de serveur EC2 de AWS différentes. Les résultats des tests montrent la performance de ces bases de données sur différents plans. Nous avons donc une idée des forces et des faiblesses de ces types de base de données. Notre test de performance est constitué de 3 charges de travail, chacune ayant une proportion d'opérations différente (uniquement en lecture, 50/50 en lecture/écriture, 10/90 lecture/écriture). Afin d'obtenir des résultats plus précis, chacune de ces charges a été exécutée 3 fois, et ce pour les deux bases de données, pour un total de 18 tests.

Cette étude nous a permis de déterminer que MongoDB est plus performant que Hyperledger Fabric. En effet, nous avons trouvé que par rapport aux latences, les opérations READ et WRITE étaient 6000 et 60 fois plus petites respectivement, ce qui nous permet de conclure que MongoDB est plus rapide à ce niveau. Ce grand écart pourrait s'expliquer par le fait que la charge de Hyperledger est significativement plus élevée que celle de MongoDB et qu'il serait possible que des erreurs aient faussé nos résultats. Il serait intéressant de refaire l'expérience en prenant en compte cette possibilité et d'adapter les tests afin d'obtenir des résultats plus exacts. Malgré cette potentielle source d'erreur, nous pouvons affirmer avec confiance que MongoDB serait tout de même plus performant que Hyperledger.

En outre, nous avons comparé les débits de chaque base de données afin d'avoir une différente perspective. L'analyse des les résultats de nos tests révèlent que MongoDB est également le plus performant à ce niveau. En effet, nous avons trouvé que les débits de MongoDB sont environ 100 à 300 fois plus performants que les débits de Hyperledger.

Dans la section des résultats, il est mentionné que MongoDB est une base de données NoSQL qui utilise des documents. Contrairement aux bases de données SQL qui parcourent à travers plusieurs tables pour trouver une donnée spécifique, une base de données comme MongoDB lit directement le document contenant les champs nécessaires pour accéder à une donnée quelconque. Ceci dit, on déduit que MongoDB est très efficace dans ses opérations. De l'autre côté, les transactions effectuées sur une base de données Hyperledger doivent passer par plusieurs étapes avant d'être approuvées. Bien qu'une base de données de type blockchain a ses avantages à ajouter ces étapes, en termes de performance, elles ne sont pas avantageuses. Nous pouvons donc conclure que MongoDB est plus performant que Hyperledger Fabric pour des opérations de lecture et d'écriture.

Toutefois, l'étude que nous avons mené présente des limites. En effet, nous ne comparons que deux bases de données; chacune représentant son type et les performances dépendent fortement de l'implémentation de celles-ci. Comme mentionné précédemment, ces résultats nous donnent une idée des forces et faiblesses de chaque type de base de données et ne constituent pas une comparaison à grande échelle. De plus, le nombre d'opérations était limité à 1000, ce qui n'est pas représentatif de l'industrie. Enfin, l'environnement utilisé pour les instances de ces bases de données étaient des machines AWS, qui peuvent présenter des variations en termes de performance d'un test à un autre. Il est important de noter que ces limites ne falsifient pas notre conclusion et que nos constats concordent avec la documentation. Il serait intéressant d'analyser les performances des bases de données blockchain et NoSQL en effectuant différents tests tel qu'un test de stress, afin d'avoir une vision encore plus claire et précise des forces et faiblesses que ces bases de données présentent.

Références

[1] J. Černiauskas, "Understanding The 4 V's Of Big Data", *Forbes*, août 2022. [En ligne]. Disponible : https://www.forbes.com/sites/forbestechcouncil/2022/08/23/understanding-the-4-vs-of-big-data/?sh=51879ce45f0a

[2] L. El-Garoui, *LOG8430 : Architectures des Megadonnées*, 2023. [En ligne]. Disponible : https://moodle.polymtl.ca/pluginfile.php/1024371/mod_resource/content/6/05.MegadonneesAnalyseEntree-H23.pdf

[3] GeeksForGeeks. (2023) Advantages of Distributed database. [En ligne]. Disponible : https://www.geeksforgeeks.org/advantages-of-distributed-database/

[4] L. El-Garoui, *LOG8430 : Blockchain*, 2023. [En ligne]. Disponible : https://moodle.polymtl.ca/pluginfile.php/1063633/mod_resource/content/5/07_Blockchain_H23.pdf

[5] MongoDB. (2023) What Is a Non-Relational Database?. [En ligne]. Disponible : https://www.mongodb.com/databases/non-relational

[6] Lourmati. (2023). Lourmati/DB-LOG8430. GitHub. [En ligne]. Disponible : https://github.com/Lourmati/DB-LOG8430

[7] Microsoft. (2022) Relational vs. NoSQL data. [En ligne]. Disponible: https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data

[8] IBM. (2022) What is Hyperledger Fabric?. [En ligne]. Disponible: https://www.ibm.com/topics/hyperledger

[9] Hyperledger Fabric. (2023) Hyperledger Fabric. [En ligne]. Disponible: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

[10] Amazon AWS. (2023) Qu'est-ce que la technologie Blockchain?. [En ligne]. Disponible: https://aws.amazon.com/what-is/blockchain/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc

[11] Amazon AWS. (2023) What is Hyperledger Fabric?. [En ligne]. Disponible: https://aws.amazon.com/blockchain/what-is-hyperledger-fabric/

[12] *X. Liang et al.*, "Integrating Blockchain for Data Sharing and Collaboration in Mobile Healthcare Applications", communication présentée à The 28th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC 2017), Montréal, Québec, Canada, Oct 2017. p. 4. [En ligne]. Disponible: https://www.researchgate.net/figure/Data-Sharing-and-Collaboration-Using-Hyperledger-Fabric-and-Channel-for-Mobile-Users_fig2_320337312

[13] MongoDB. (2023) What is a Document Database?. [En ligne]. Disponible: https://www.mongodb.com/document-databases

[14] MongoDB. (2023). MongoDB Architecture Guide. [En ligne]. Disponible: https://www.mongodb.com/collateral/mongodb-architecture-guide

[15] Sean Busbey. (2020). Core YCSB properties. [En ligne]. Disponible: https://github.com/brianfrankcooper/YCSB/wiki/Core-Properties

[16] Williams, Alex. (2021) NoSQL database types explained: Document-based databases. [En ligne]. Disponible: https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Document-based-databases