# Architecture d'Analyse de Big Data Basée sur l'Apprentissage Profond pour le Stationnement Intelligent: Étude et Comparaison des Outils

Samira Yessad, Souaad Boussoufa-Lahlah, Wissam Ferroudj, Kenza Djellaoui, Louiza Bouallouche-Medjkoune

# Architecture d'Analyse de Big Data Basée sur l'Apprentissage Profond pour le Stationnement Intelligent : Étude et Comparaison des Outils

Samira Yessad and Souaad Boussoufa-Lahlah
*Unité de recherche LaMOS*
*Faculté des Sciences Exactes, Université de Bejaia*
06000 Bejaia, Algérie
{samira.yessad & souaad.lahlah}@univ-bejaia.dz

Wissam Ferroudj and Kenza Djellaoui
*Département d'Informatique*
*Université de Bejaia*
06000 Bejaia, Algérie
{wissam.ferroudj & kenza.djellaoui}@se.univ-bejaia.dz

Louiza Bouallouche-Medjkoune
*Unité de recherche LaMOS*
*Faculté des Sciences Exactes, Université de Bejaia*
06000 Bejaia, Algérie
louiza.medjkoune@univ-bejaia.dz

***Résumé*—L'émergence de différents appareils miniaturisés de l'Internet des objets et leur forte utilisation dans les applications de villes intelligentes génèrent avec une grande vitesse un grand volume de différentes données connues sous le concept de Big Data (ou mégadonnées). Ces données ne peuvent être utiles que si nous les analysons avec des outils performants, soit pour tirer des statistiques afin de savoir ce qui s'est passé, pourquoi et comment, soit pour prédire ce qui va se passer et faciliter la prise de décision. Dans ce travail, nous avons proposé une nouvelle architecture d'analyse de Big Data basée sur l'apprentissage profond en lien avec les applications de stationnement intelligent, avec une étude comparative de trois grandes technologies dédiées au Big Data qui sont Hadoop, Spark et Flink, dans laquelle nous avons discuté de leurs architectures, fonctionnements et fonctionnalités. Cette étude nous permettra de choisir les outils appropriés à l'implémentation de notre système de stationnement intelligent.**

***Mots Clés*—Architecture d'analyse de Big Data, Plateformes Big Data (Hadoop, Spark et Flink), Apprentissage profond, Stationnement intelligent.**

## I. Introduction

Une ville intelligente est tout un environnement connecté dont l'objectif est de mieux utiliser les ressources publiques afin de réduire les coûts de fonctionnement des administrations publiques, et ainsi améliorer la qualité des services offerts à ses citoyens pour leur assurer une vie meilleure et confortable. Elle s'appuie sur l'intégration des Technologies de l'Information et de la Communication (TIC), des capteurs et de l'Internet des Objets (IdO) dans les divers secteurs de la ville. Ces derniers génèrent à grande vitesse une grande quantité de données hétérogènes, qui est considérée comme le nouvel or noir du XXIe siècle. Connue beaucoup plus sous le nom de Big Data (BD), cette précieuse richesse a suscité l'intérêt de beaucoup de chercheurs, d'industriels et d'investisseurs. En effet, les applications BD, comme celles par exemple dédiées aux villes intelligentes, génèrent des données de manière continue à partir de différentes sources. Ainsi, les données ont diverses caractéristiques résumées dans les 10 Vs du Big Data [1] (le Volume, la Vitesse, la Variété, la Valeur, la Véracité, la Visualisation, la Validité, la Volatilité, la Viabilité et la Viscosité) [2]. Le taux de génération de données varie selon les capteurs utilisés et en fonction des exigences de l'application (les soins et la santé, la gestion de l'énergie, le marketing, les services clients, les transports, le développement durable, l'éducation, le stationnement intelligent, etc.). Par exemple, les capteurs GPS sont capables de générer des données dans des intervalles de temps très réduits d'environ quelques secondes, tandis que les capteurs de températures peuvent générer des données dans des intervalles distants d'environ une heure [3]. Cette hétérogénéité et ce grand volume de données ont fait qu'aucun outil traditionnel de gestion des bases de données ne peut vraiment gérer le BD, d'où l'émergence de nouvelles technologies permettant de stocker (MongoDB [4], Redis [5], Casandra [6], Hbase [6], Neo4j [7], etc.), de traiter et d'exploiter efficacement ce type de donnée (Hadoop, Spark, Flink, Storm, Kafka, etc [6].).

Dans ce travail, nous nous intéressons aux applications de stationnement intelligent qui, comme les autres applications des villes intelligentes, peuvent tirer profit de cet or noir en développant une architecture efficace de l'analyse de BD. Ainsi et dans la perspective de contribuer à atteindre cet objectif, nous proposons une nouvelle architecture d'analyse de BD basée sur l'apprentissage profond pour améliorer le stationnement intelligent. Toutefois, pour implémenter et mettre en œuvre cette dernière, nous nous sommes retrouvés face à une multitude de choix de technologies de BD, et afin de faire un bon choix, nous avons commencé par réaliser une étude comparative de ces dernières.

Cet article commence par une introduction suivie de la Section II qui présente les métriques à considérer lors du choix

d'un outil ou d'une plateforme BD. Ensuite, nous présentons dans la Section III un état de l'art de quelques architectures d'analyse de BD pour les systèmes de stationnement intelligent existantes dans la littérature, ainsi que l'architecture que nous proposons. Puis, nous enchaînons avec les Sections IV et V qui présentent respectivement une étude comparative des infrastructures de stockage et plateformes de traitement et d'analyse de BD. Nous terminons par une conclusion et des perspectives.

## II. Choix des outils/plateformes BD

Lors du choix d'une plateforme pour le traitement des données, il est essentiel d'examiner les besoins de l'application concernée. Pour ce faire, l'utilisateur doit se poser quelques questions fondamentales, sur par exemple, les délais de traitement et la taille des données à traiter avant de prendre une décision éclairée [8]. Dans ce cadre, plusieurs métriques et critères entrent en jeu.

### A. Scalabilité

La scalabilité fait référence à la capacité d'un système à fonctionner efficacement lorsqu'il est soumis à une augmentation de la charge de travail. En d'autres termes, un système doit être en mesure de gérer une augmentation du nombre d'utilisateurs, de transactions ou de données sans que cela n'affecte négativement ses performances. Pour évaluer la scalabilité d'un outil, il est important de considérer des facteurs tels que la capacité de traitement, la capacité de stockage, la bande passante et la disponibilité des ressources.

### B. La tolérance aux pannes

La tolérance aux pannes fait référence à la capacité d'un système à continuer de fonctionner même en présence de pannes matérielles ou logicielles. Pour évaluer la tolérance aux pannes d'un outil ou d'une plateforme, il est important de considérer des facteurs tels que la redondance, les mécanismes de récupération, la détection des pannes et les mécanismes de réparation automatique.

### C. Modes de traitement des mégadonnées

Différentes plateformes de traitement sont utilisées pour le traitement de gros volumes de données provenant de différents capteurs. Selon les besoins en matière de données, les infrastructures informatiques peuvent traiter les données en mode batch (en lot), en temps quasi réel ou en temps réel [9].

*a) Mode batch ou traitement par lot :* c'est un mode de traitement des mégadonnées dans lequel les données sont collectées, stockées et traitées en blocs. Les données sont collectées sur une période donnée, puis traitées en une seule opération. Les traitements en batch sont souvent utilisés pour les opérations de traitement lourdes qui prennent du temps et des ressources, tels que les rapports, les analyses ou la génération de statistiques.

*b) Traitement en temps quasi réel/en continu (stream processing en anglais) :* c'est le traitement ou l'analyse de flux continu d'événements. Les données sont traitées à la volée, sans avoir besoin d'attendre que toutes les données soient collectées avant de les traiter. Ce mode de traitement est souvent utilisé pour traiter de grands volumes de données en temps proche au réel.

*c) Traitement en temps réel :* c'est un mode de traitement qui permet de traiter les données au moment où elles sont créées ou reçues, sans délai perceptible. Ce qui signifie que le temps de réponse pour le traitement est très court. Il est souvent utilisé dans des applications en temps réel de surveillance, de contrôle de processus, de détection de fraude, de transactions financières, etc. À la différence d'un stream processing qui s'occupe de la vélocité de données, le traitement temps réel s'occupe de la latence des données [10].

### D. Taille des données supportées

La taille des données supportées est la capacité d'une plateforme à gérer des fichiers de données de grande taille, souvent supérieur à des gigaoctets. Les plateformes doivent être capables de stocker et de traiter de grandes quantités de données, en évitant les goulets d'étranglement liés aux limitations de taille de fichier ou de stockage. Les plateformes qui offrent une haute capacité de stockage et des mécanismes de partitionnement de données sont souvent préférées.

### E. Sécurité

La sécurité est essentielle lors du traitement de données sensibles. Il est important de choisir une solution qui offre des fonctionnalités de sécurité robustes pour protéger les données.

## III. État de l'art sur les architectures d'analyse de Big Data pour le stationnement intelligent et nouvelle proposition

Plusieurs travaux sur le stationnement intelligent existent dans la littérature, certains proposent une architecture pour l'analyse du Big Data et d'autres travaillent directement sur un module de l'architecture sans la spécifier. Dans cette section, nous donnerons un bref aperçu des travaux qui ont présenté explicitement l'architecture d'analyse de données sur laquelle se base leurs systèmes de stationnement intelligents, puis présenterons celle que nous proposons dans la Fig. 1.

Dans [11] et [12], une architecture pour l'analyse des BD de l'IdO pour la ville intelligente est proposée, elle inclut, parmi les applications, le stationnement intelligent. La première [11] est une architecture de quatre couches ; génération et collecte de données, communication, gestion et traitement de donnée et interprétation des données. Pour la gestion et le traitement de données, les auteurs proposent l'utilisation de l'eco-système Hadoop avec Spark, Storm [6] ou VoltDB [13] pour le traitement de flux en temps réel, MapReduce pour l'analyse et HDFS, HBASE, HIVE [6] et SQL pour le stockage de Big Data. Dans [12], on trouve une architecture composée de la source de données basée sur l'IdO qui envoie les données à travers des passerelles via Internet vers le composant de

construction de la ville intelligente qui se compose à son tour d'une couche de pré traitement de données (collection, filtrage et classement des données), une couche de traitement de données qui utilise Hadoop pour le traitement en lot et Spark pour le traitement temps réel, et enfin une couche analyse de données qui utilise les différentes bibliothèques comme l'apprentissage automatique et les modèles de décision. Le dernier composant est celui des applications qui peuvent utiliser les résultats de l'analyse des données.

Dans [14], les auteurs ont présenté leur architecture qui est composée de quatre couches, elle est inspirée de celle de [11]. La plus basse est la couche de génération et de collecte de données, elle s'occupe de la collecte et de la combinaison des données perçues par les appareils de l'IdO ainsi que de leur filtrage préliminaire. La deuxième couche, qui est la première couche intermédiaire, est responsable de la communication des données. La troisième couche est la deuxième couche intermédiaire qui est la couche centrale de l'ensemble du système analytique et est responsable du traitement des données en utilisant les mêmes outils proposés dans [11], à savoir, Hadoop Spark, Storm, VoltDb, HIVE, HBASE et SQL. La couche supérieure est celle d'interprétation des données qui représente l'application d'exploitation des données par l'utilisateur final.

Les auteurs de [15] ont classé les principales composantes du système de stationnement intelligent en plusieurs catégories, l'infrastructure où résident la puissance de calcul et les données, l'analyse où se trouve l'intelligence, et le tableau de bord où tout, par le biais de la visualisation, reçoit le contexte et la signification appropriés. Ainsi, leur architecture a quatre couches, l'ingestion de données, le centre de données où sont stockées les données brutes, également appelé lac de données (Data Lake) et qui font objet de différents types d'analyses de big data, temps réel et avec l'apprentissage automatique, la visualisation des données en utilisant les méthodes d'apprentissage automatique et enfin la couche d'accès au données qui peut se faire par application mobile, par web ou un tableau de bord en ligne. L'idée principale dans ce travail est d'inclure l'intelligence artificielle, l'apprentissage automatique et les plateformes d'analyse de BD dans un système de stationnement intelligent.

L'architecture du système de stationnement intelligent basé sur l'IdO proposé dans [16] comporte quatre couches. La couche capteurs qui s'occupe de la collecte des informations à partir de différents emplacements de stationnement grâce à divers types de capteurs. La couche communication est la couche intermédiaire qui permet de transmettre les informations collectées vers le service en nuage (le cloud) qui constitue la couche traitement. Cette dernière stocke et analyse un grand volume de données et elle est connectée à la dernière couche qui est la couche services qui offre des applications de gestion de véhicules et de parkings. Les auteurs de ce travail comme le précédent proposent un système de prise de décisions basé sur l'apprentissage profond pour la prédiction de la disponibilité des espaces de parking sans utiliser les technologies du BD pour le traitement et l'analyse de données et l'implémentation a été faite en utilisant la plateforme Keras.

Une application mobile qui notifie les conducteurs sur leurs mobiles de la disponibilité d'espace de stationnement est proposée dans [17]. C'est un système de prédiction basé sur l'apprentissage profond mobile qui en fonction de la position du conducteur et les données récupérées de la ville intelligente en temps réel, prédit si une place de stationnement est disponible pour le conducteur ou pas. L'implémentation du système est basée sur Keras et TensorFlow.

Compte tenu de l'étude bibliographique que nous avons effectuée, nous proposons une nouvelle architecture d'analyse de Big Data basée sur l'apprentissage profond pour le stationnement intelligent comme présentée dans la Fig. 1.

Elle intègre le traitement des données en lot, en flux continu et en temps réel pour pouvoir effectuer une analyse descriptive et prédictive. La première peut donner l'occupation des places de stationnement à différentes heures de pointe de la journée pour aider les autorités de la ville à élargir des parkings ou d'en ouvrir d'autres à proximité. Elle peut aussi aider à constater la demande de stationnement dans différentes zones lors d'événements spéciaux et guider les conducteurs pour qu'ils trouvent un parking approprié rapidement et réduire les embouteillages dans la ville. En plus, l'analyse des valeurs moyennes d'occupation de chaque parking à chaque heure d'un jour donné peut fournir aux automobilistes plus d'informations sur le stationnement à l'échelle horaire, ce qui pourrait les aider à prendre une décision de stationnement en fonction de l'heure à laquelle ils veulent entrer dans la ville.

D'un autre coté, une analyse prédictive peut estimer le temps d'arrivée du conducteur en fonction duquel on peut prédire si une place de stationnement dans un parking reste vacante à son arrivée. En plus, on peut utiliser la détection de l'occupation pour fournir des informations en temps réel sur la disponibilité des places de stationnement et procéder à

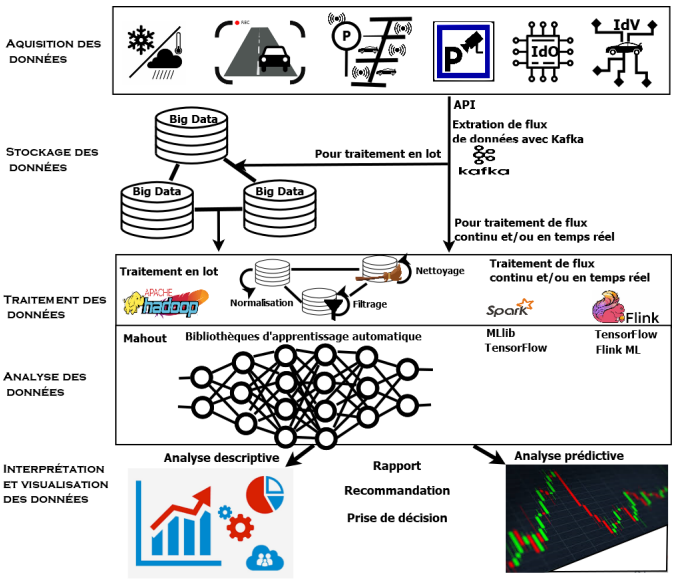


FIGURE 1. Architecture d'analyse de Big Data basée sur l'apprentissage profond pour le stationnement intelligent.

des ajustements de prix et les limites de temps en fonction de la demande pour une utilisation efficace des places de stationnement dans une zone encombrée, etc [18] [19].

Pour passer à l'implémentation de cette architecture, nous avons besoin de faire une étude comparative des outils de stockage et d'analyse de Big Data pour faire un bon choix et développer un système efficace. Ceci est l'objectif des sections suivantes.

## IV. Infrastructure de stockage

Dans les villes intelligentes, les données collectées sont très diverses et peuvent prendre différentes formes, allant du texte brut aux données multimédias. Par conséquent, en plus des systèmes de stockage traditionnels basés sur une représentation structurée des données sous forme de relations et utilisant SQL, les villes intelligentes ont également besoin de structures de stockage plus avancées capables de stocker des données semi-structurées et non structurées [20] connus sous le nom NoSQL (pour Not Only SQL). De ce fait, il existe plusieurs types de bases de données dédiées aux Big Data, chacune ayant ses propres caractéristiques et avantages [21] [22]. Voici une comparaison de trois bases de données Big Data les plus populaires et que nous jugeons plus adaptées pour notre système de stationnement intelligent [23] [24].

### A. MongoDB

MongoDB [4] est une base de données multiplateforme orientée document. Elle offre des performances élevées et est de nature dynamique où nous n'avons pas besoin de prédéfinir un schéma pour les données. MongoDB stocke les données au format JSON, ce qui facilite l'envoi des données sous la forme désirée, et elle offre une vitesse, une disponibilité et une évolutivité élevées. Elle est considérée comme la meilleure base de données NoSQL pour gérer l'analyse de données en temps réel [24].

### B. Cassandra

Cassandra [6] est un système de gestion de base de données open source, caractérisé par une architecture distribuée capable de gérer une grande quantité de données sur différents serveurs de base, et offre une haute disponibilité sans aucune sorte de panne. Il est écrit en Java et développé par Apache Software Foundation. Il dispose d'un système distribué peer-to-peer sur ses nœuds, ainsi les données sont réparties entre tous les nœuds du cluster.

### C. HBase

C'est une base de données distribuée open source développée par les fondations logicielles Apache [6], le fruit d'une amélioration de Google Bigtable. Cette plateforme principalement écrite en java, peut stocker des quantités massives de données allant de téraoctets à pétaoctets sous forme de tableaux [6]. Sous Hadoop, HBase est utilisée pour stocker les données massives au niveau de HDFS. Un des avantages de cette plateforme est de fournir un accès aléatoire et rapide à une grande quantité de données structurées. HBase est utilisée pour l'analyse des journaux en ligne, les distributions Hadoop, les applications à forte écriture (applications lourdes) et MapReduce.

### D. Comparaison entre les bases de données NoSQL, MongoDB, Cassandra et Hbase

Nous avons mené une riche recherche bibliographique pour comprendre les caractéristiques de ces trois bases de données et présenter une étude comparative entre elles selon quelques critères que nous présentons dans le Tableau I.

## V. Plateformes de traitement et d'analyse de données

Il s'agit de l'ensemble des plateformes analytiques de Big Data qui servent d'interface pour collecter les données, effectuer les analyses et les traitements requis en utilisant des modes de traitement de données appropriés. Nous avons effectué une recherche bibliographique pour comprendre et comparer les trois plus grandes technologies de ce domaine qui sont les plus utilisées à savoir Hadoop, Spark et Flink [28]. Nous présentons ainsi une comparaison de ces trois technologies considérant quelques critères tel que présenté dans le Tableau II.

### A. Hadoop

Hadoop est une plateforme open source qui fait partie d'une fondation Apache basée sur Java. Elle permet de gérer et traiter de grands ensembles de données, structurées et non structurées, qui résident sous la forme de clusters, dans le cadre d'un système distribué. Cette plateforme informatique a été conçue afin de résoudre les problèmes liés à la volumétrie et à la variété des données. Elle offre un espace de stockage massif pour tous les types de données, une immense puissance de traitement ainsi que la possibilité de prendre en charge une quantité de tâches virtuellement illimitées [6]. Parmi les sociétés géantes qui utilisent Hadoop on trouve Twitter, LinkedIn, ou encore eBay et Amazon.

### B. Spark : 3G de BigData

Spark est un moteur open-source conçu en 2009 afin d'accélérer le traitement des systèmes Hadoop. Il permet une analyse et un traitement ultra-rapide de Big Data en cluster. Son utilisation est principalement dédiée aux applications du Machine Learning et des pipelines de données. Parmi les grandes sociétés du monde de l'internet qui ont développé Spark à très grande échelle, on trouve Netflix, Yahoo et ebay. Il est idéal pour des informations immédiates en temps réel et un accès aux données existantes.

### C. Flink : 4G de Big Data

Flink est un moteur de traitement distribué et une plateforme open-source d'analyse de données évolutif. Il est conçu pour répondre aux problématiques de traitement de données par lots grâce à la combinaison du traitement en streaming (en mémoire) et du traitement par lot (sur disque) [29] avec une gestion automatique de la mémoire. Le streaming Flink traite les flux de données comme de vrais flux. Dès que les données arrivent elles sont immédiatement "transférées"

TABLE I
ANALYSE ET COMPARAISON DES BASES DE DONNÉES NOSQL, MONGODB, CASSANDRA ET HBASE

| | Hbase | MongoDB | Cassandra |
|---|---|---|---|
| Modèle de base de données | Orienté colonnes | Orienté document | Orienté colonnes |
| Réplication [21] [22] | Maître-esclave | Maître-esclave | Maître-esclave |
| Langage de programmation code/ pris en charge | Java/ C, C#, C++, Groovy, Java, PHP, Python, Scala | C++/ C, C#, C++, Erlang, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, Scala | Java/ C#, C++, Clojure, Erlang, Go, Haskell, Java, Node.js, Perl, PHP, Python, Ruby, Scala |
| Latence [24] [25] [26] | Temps d'exécution élevé pour les lectures et réduit pour les mises à jour | Temps d'exécution moyen pour les lectures et les mises à jour | Temps d'exécution moyen pour les lectures et réduit pour les mises à jour |
| Langage de requêtes [22] | API calls, REST, XML, Thrift | Système de fichiers à mémoire volatile | CQL, API calls, Thrift |
| Cohérence | Forte cohérence | Forte cohérence | Éventuelle cohérence. |
| Lecture et écriture | Bon pour les lectures intensives, écrit sur un seul serveur de données | L'évolutivité d'écriture est limitée. Les performances de lecture sont moins rapides que dans Cassandra. | Douée pour l'écriture, écrit sur plusieurs serveurs avec des versions différentes |
| Les systèmes d'exploitation du serveurs | Linux, Unix, Windows | Solaris, Linux, OS X, Windows | BSD, Linux, OS X, Windows |
| Sécurité [27] | Authentification : moyen, Contrôle d'accès : moyen, Chiffrement : faible, Vérification : moyen | Authentification : moyen, Contrôle d'accès : élevé, Chiffrement : moyen, Vérification : faible | Authentification : faible, Contrôle d'accès : faible, Chiffrement : moyen, Vérification : faible |

via un programme de streaming. Il est également capable de gérer les données tardives dans les flux en utilisant des filigranes. De plus, Flink fournit un mode de compatibilité très puissant qui permet d'utiliser le code Storm, MapReduce, ... existant sur le moteur d'exécution Flink. Contrairement à Spark, Flink est indépendant de Hadoop mais il peut rejoindre d'autres infrastructures de l'écosystème Hadoop, telles que Spark, Storm et Samza, [6] dans la course à la gestion en flux du Big Data ainsi qu'à l'utilisation de HDFS (Hadoop Distributed File System) pour lire, écrire, stocker et traiter les données [30].

### *D. Comparaison entre Spark, Hadoop et Flink*

Le Tableau II illustre une comparaison des outils de traitement de Big Data Hadoop, Spark et Flink [30].

## VI. CONCLUSION

Il est primordial de bâtir une bonne architecture technique appropriée sur laquelle les données seront collectées, stockées, traitées et analysées en temps réel. Cette architecture doit être à la fois résistante aux pannes, puissante et surtout évolutive. Car c'est le choix de la bonne architecture qui aide à la bonne exploitation des grands flux de données qui voient le jour à chaque instant, et qui à son tour mène automatiquement à la réalisation d'importantes valeurs ajoutées à long terme sur le plan économique de la ville et pour le confort des citoyens. Pour ce qui est de notre architecture pour le stationnement intelligent, nous avons commencé par développer un modèle de prédiction des espaces de stationnement libres en tirant profit des données stockées dans différents datasets en comparant plusieurs modèles d'apprentissage profond. Pour accélérer l'analyse et le traitement des données dans notre système, nous envisageons de distribuer le modèle d'apprentissage le plus performant en se basant sur l'un des trois outils étudiés dans ce travail. Compte tenu de l'étude effectuée dans ce travail, nous constatons que Flink et Spark peuvent avoir des performances très proches et le choix entre les deux est dépendant de l'application et d'autres paramètres. Ainsi, nous devons effectuer une analyse expérimentale du modèle distribué sur différentes plateformes avec différentes bibliothèques d'apprentissage profond [31] (Spark avec BigDL [32], Flink avec TensoFlow [33] , ...) et choisir les outils de stockage, de traitement et d'analyse les plus appropriés pour l'implémentation de notre système de stationnement intelligent.

## RÉFÉRENCES


[1] N. Khan, M. Alsaqer, H. Shah, G. Badsha, A. A. Abbasi, and S. Salehian, "The 10 Vs, issues and challenges of big data," in *Proceedings of the 2018 international conference on big data and education*, pp. 52–56, 2018.

[2] L. A. Schintler and C. L. McNeely, *Encyclopedia of big data*. Springer, 2022.

[3] D. Pal, T. Triyason, and P. Padungweang, "Big data in smart-cities : Current research and challenges," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 6, no. 4, pp. 351–360, 2018.

[4] "MangoDB." https ://www.mongodb.com/. Dernier accès le 30-04-2023.

[5] "Redis." https ://redis.io/. Dernier accès le 01-04-2023.

[6] "The Apache Software Foundation." https ://www.apache.org/. Dernier accès le 22-05-2023.

[7] "Neo4j." https ://neo4j.com/fr/. Dernier accès le 01-04-2023.

[8] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," *Journal of big data*, vol. 2, no. 1, pp. 1–20, 2015.

[9] M. Goudarzi, "Heterogeneous architectures for big data batch processing in MapReduce paradigm," *IEEE Transactions on Big Data*, vol. 5, no. 1, pp. 18–33, 2019.

[10] N. Tantalaki, S. Souravlas, and M. Roumeliotis, "A review on big data real-time stream processing and its scheduling techniques," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 5, pp. 571–601, 2020.

[11] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, "Urban planning and building smart cities based on the internet of things using big data analytics," *Computer networks*, vol. 101, pp. 63–80, 2016.

[12] M. M. Rathore, A. Paul, W.-H. Hong, H. Seo, I. Awan, and S. Saeed, "Exploiting IoT and big data analytics : Defining smart digital city using real-time urban data," *Sustainable cities and society*, vol. 40, pp. 600–610, 2018.

TABLE II
COMPARAISON DES OUTILS PILIERS DE L'ANALYSE BIG DATA

| Fonctionnalité | Spark | Hadoop | Flink |
|---|---|---|---|
| Scalabilité | Très scalable en utilisant la structure de données RDD (Resilient Distributed Dataset) distribuée sur plusieurs nœuds, l'optimisation des opérations en mémoire et la minimisation des E/S disque. | Une grande scalabilité grâce à HDFS qui permet le parallélisme et à son modèle de calcul MapReduce | Une grande scalabilité, parallélise les opérations sur les flux en les distribuant sur plusieurs nœuds pour s'adapter à des volumes de données importants et à des charges de travail croissantes. |
| Tolérances aux pannes | Obtenue en stockant la chaîne de transformations | Haute et obtenue en répliquant des blocs de données | Basée sur les instantanés distribués de Chandy-Lamport pour un débit élevé. |
| Mode de traitement | Traitement par lots et en streaming | Traitement par lots | Traitement en streaming et par lots |
| Taille des données supportées | Pétaoctets à exaoctets | Pétaoctets à exaoctets | Gigaoctets à pétaoctets et au-delà |
| Coût | Coûteux en mémoire et en nombre de clusters | Moins coûteux | Coûteux en mémoire et en nombre de clusters |
| Sécurité | Sécurisé grâce à l'intégration avec Hadoop | Extrêmement sécurisé via l'authentification, l'autorisation, le chiffrement, l'audit des accès et l'intégration avec par exemple Kerberos | Support d'authentification des utilisateurs via l'authentification, l'autorisation et le chiffrement des données. |
| Rapidité de traitement et performances | Traitement rapide grâce au stockage des données intermédiaires en mémoire. | Traitement ralenti à cause de la lecture et écriture à partir d'un disque | Le plus rapide grâce aux opérateurs d'itération en boucle fermée natifs qui accélèrent l'apprentissage automatique et le traitement des graphes. |
| Simplicité d'utilisation | Facile grâce à ses opérateurs de haut niveau | Difficile à cause de codage de chaque opération à la main | Facile grâce à ses opérateurs de haut niveau |
| Gestion de la mémoire | Gestion configurable | Gestion configurable | Gestion automatique |
| Language pris en charge | Java, R, Scala, Python | Java, c++, Ruby, Groovy, Perl, Python. | Java, Scala, Python et R |
| Les bibliothèques ML | MLlib | Mahoot | Flink ML |


[13] "VoltDB." https ://www.voltactivedata.com/. Dernier accès le 12-04-2023.

[14] R. S. Cokro, E. Y. Wirawan, Y. Putra, A. Puspitarini, G. Wang, and E. R. Kaburuan, "Designing smart parking system through the use of IoT and big data," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 10, no. 5, pp. 3023–3027, 2021.

[15] M. M. Yusof, S. M. Salleh, N. A. M. Sari, and R. A. Rahim, "Preliminary technical insights in utilizing big data architecture and analytics for the implementation of smart parking dashboarding," *International Journal of Technology Management and Information System*, vol. 4, no. 4, pp. 66–74, 2022.

[16] G. Ali, T. Ali, M. Irfan, U. Draz, M. Sohail, A. Glowacz, M. Sulowicz, R. Mielnik, Z. B. Faheem, and C. Martis, "IoT based smart parking system using deep long short memory network," *Electronics*, vol. 9, no. 10, p. 1696, 2020.

[17] M. T. Rahman, Y. Zhang, S. A. Arani, and W. Shao, "MDLpark : Available parking prediction for smart parking through mobile deep learning," in *Wireless Sensor Networks : 16th China Conference, CWSN 2022, Guangzhou, China, November 10–13, 2022, Proceedings*, pp. 182–199, Springer, 2022.

[18] S. Nguyen, Z. Salcic, and X. Zhang, "Big data processing in fog-smart parking case study," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 127–134, IEEE, 2018.

[19] R. K. Lenka, R. K. Barik, N. K. Das, K. Agarwal, D. Mohanty, and S. Vipsita, "PSPS : An IoT based predictive smart parking system," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, pp. 311–317, IEEE, 2017.

[20] S. Sharma, "Expanded cloud plumes hiding big data ecosystem," *Future Generation Computer Systems*, vol. 59, pp. 63–92, 2016.

[21] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Comparison and classification of NoSQL databases for big data," *International Journal of Database Theory and Application*, vol. 6, no. 4.2013, 2013.

[22] A. Moniruzzaman and S. A. Hossain, "NoSQL database : New era of databases for big data analytics-classification, characteristics and comparison," *arXiv preprint arXiv :1307.0191*, 2013.

[23] "DB-Engines Ranking." https ://db-engines.com/en/ranking. Dernier accès le 01-04-2023.

[24] A. Gandini, M. Gribaudo, W. J. Knottenbelt, R. Osman, and P. Piazzolla, "Performance evaluation of NoSQL databases," in *Computer Performance Engineering : 11th European Workshop, EPEW 2014, Florence, Italy, September 11-12, 2014. Proceedings 11*, pp. 16–29, Springer, 2014.

[25] V. Abramova, J. Bernardino, and P. Furtado, "Experimental evaluation of NoSQL databases," *International Journal of Database Management Systems*, vol. 6, no. 3, p. 1, 2014.

[26] H. Matallah, G. Belalem, and K. Bouamrane, "Evaluation of NoSQL databases : MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB," *International Journal of Software Science and Computational Intelligence (IJSSCI)*, vol. 12, no. 4, pp. 71–91, 2020.

[27] A. Zahid, R. Masood, and M. A. Shibli, "Security of sharded NoSQL databases : A comparative analysis," in *2014 conference on information assurance and cyber security (CIACS)*, pp. 1–8, IEEE, 2014.

[28] F. Ullah, S. Dhingra, X. Xia, and M. A. Babar, "Evaluation of distributed data processing frameworks in hybrid clouds," *arXiv preprint arXiv :2201.01948*, 2022.

[29] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache Flink : Stream and batch processing in a single engine," *The Bulletin of the Technical Committee on Data Engineering*, vol. 38, no. 4, 2015.

[30] E. Nazari, M. H. Shahriari, and H. Tabesh, "BigData analysis in healthcare : Apache Hadoop, Apache Spark and Apache Flink," *Frontiers in Health Informatics*, vol. 8, no. 1, p. 14, 2019.

[31] K. Kumar, N. A. Sharma, and A. S. Ali, "Machine learning solutions for investigating streams data using distributed frameworks : Literature review," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pp. 1–6, IEEE, 2021.

[32] J. J. Dai, Y. Wang, X. Qiu, D. Ding, Y. Zhang, Y. Wang, X. Jia, C. L. Zhang, Y. Wan, Z. Li, *et al.*, "BigDL : A distributed deep learning framework for big data," in *Proceedings of the ACM Symposium on Cloud Computing*, pp. 50–60, 2019.

[33] S. Horchidan, E. Kritharakis, V. Kalavri, and P. Carbone, "Evaluating model serving strategies over streaming data," in *Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, pp. 1–5, 2022.