République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de 8 Mai 1945-Guelma-
Faculté des Mathématique,d'Informatique et des Sciences de la matière
Département d'Informatique

Mémoire de fin d'études Master

Filiere : Informatique

Option :

Système Informatique

Thème

Optimisation de la structure d'indexation de l'arbre BCCF pour des données hétérogènes dans un environnement Edge/Fog/Cloud Computing

Encadré par :
D^r Kouahla Zinedine

Présenté par :
M^me kemouguette Ibtissem

Membres de jury :
D^r Djakhdjakha Lynda
D^r Soussi Hakim

Année Universitaire 2020/2021

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

Je tiens à remercier tout particulièrement mon encadrant **Dr.Kouahla Zineddine**, Pour l'aide compétente qu'il m'a apportée, pour sa patience et ses encouragements. Son œil critique m'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Je tiens à remercier les membres du jury **Dr. Djakhdjakha Lynda** et **Dr. Soussi Hakim** pour leur présence, leur lecture attentive de mon mémoire ainsi que pour les remarques qu'ils m'adresseront lors de la soutenance afin d'améliorer ce manuscrit.

je tiens à remercier également le doctorant **Benrazek Ala-Eddine** pour son encouragement, son aide et ses conseils qui m'ont beaucoup aidé surtout pour la maîtrise de la recherche pour finaliser ce travail.

Un grand merci pour le laboratoire **LABSTIC** et tous ses membres, c'était vraiment une très bonne expérience d'avoir passé un stage dans ce laboratoire.

Enfin, j'adresse de tendres remerciements à toute ma famille, mes parents, mes frères, mon mari, mes amis et toute personne ayant contribué de prés ou de loin à la réalisation de ce travail.

Résumé

Ces dernières années, avec le développement de la technologie et l'énorme quantité de données hétérogènes générées par les capteurs de l'Internet des objets (IoT), la plupart des systèmes traditionnels de traitement des bases de données ne sont pas câblés pour traiter et stocker cette grande quantité de données, ce qui a obligé les chercheurs à concevoir de nouveaux systèmes basés sur des mécanismes d'indexation.

Parmi les dernières structures d'indexation dans l'espace métrique, nous trouvons l'arbre BCCF. Cette structure est basée sur un partitionnement récursif de l'espace en utilisant l'algorithme de regroupement k-means. Malgré les avantages offerts par cette solution, elle reste confrontée au problème du chevauchement des données entre ses nœuds, en particulier dans le cas de données massives, ce qui a un impact négatif sur la qualité de l'index, ainsi que sur les performances du processus de recherche. Cette étude vise à optimiser la structure du BCCF pour le rendre capable de traiter des données massives de l'IdO. Pour cela, nous proposons l'utilisation du regroupement comme prétraitement , pour minimiser le coût de construction et améliorer la qualité du processus de découverte et de recherche des données massives d'IdO.

Mots clés : Indexation, Internet des Objets, Optimisation, Recherche Knn, Similarité.

Abstract

In recent years, with the development of technology and the huge amount of heterogeneous data generated by Internet of Things (IoT) sensors, most traditional database processing systems are not wired to process and store this large amount of data, requiring researchers to design new systems based on indexing mechanisms.

Among the latest indexing structures in metric space is the BCCF tree. This structure is based on a recursive partitioning of the space using the k-means clustering algorithm. Despite the advantages offered by this solution, it still faces the problem of overlapping data between its nodes, especially in massive data, which has a negative impact on the quality of the index, as well as on the performance of the search process. This study aims to optimize the structure of the BCCF to make it capable of handling massive IoT data. For this purpose, we duly propose the use of clustering as a pre-processing , to minimize the construction cost and improve the quality of the discovery and search process of massive IoT data.

Keywords : Indexing, Internet of Thigs, Optimization, Knn search, Similarity.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Comme le titre de ce mémoire l'indique, dans ce travail, nous nous sommes intéressés à l'indexation de données dans un espace métrique. Ce besoin est né de la nécessité d'indexer et de rechercher des données de plus en plus nombreuses et complexes.

Introduisons donc le contexte général de ce travail, ainsi que les problèmes à résoudre dans ce mémoire.

Contexte et Problématique

l'un des aspects les plus importants de l'Internet des Objets est qu'il transforme des objets simples en objets intelligents capables de générer et transférer des données sur un réseau et de communiquer entre eux sans interaction humaine [1]. Tout cela contribue à améliorer la vie d'une part et à augmenter le nombre d'objets connectés et de données générées d'autre part. Au fil des années, la quantité de données a augmenté de façon exponentielle. Selon les statistiques publiées par le « Statista Research Department » [2], d'ici à 2025, les prévisions suggèrent qu'il y aura plus de 75 milliards d'objets connectés à l'Internet en service. Cela représenterait une augmentation de presque trois fois par rapport à l'année 2019.

Cette expansion a créé le besoin de stocker, gérer et sécuriser d'énormes volumes de données afin de tirer le meilleur parti de cette masse d'informations pour un public toujours plus nombreux. Pour relever ces défis, un nouveau paradigme informatique appelé Cloud computing est apparu comme une solution prometteuse, en raison de sa capacité (théoriquement) illimitée de stockage et de traitement, de sa disponibilité, de son indépendance géographique et de la transparence des données et des services.

Néanmoins, le stockage et le traitement de ces données dans le Cloud ne sont pas pratiques pour les applications en temps réel telles que la vidéosurveillance, le suivi des patients, les systèmes de contrôle des véhicules, etc. en raison de leurs exigences de latence. Par conséquent, une nouvelle solution est nécessaire pour résoudre ce problème. Pour cela, le Fog (**F**rom c**O**re to ed**G**e) computing est apparu [3, 4] comme une solution fondée sur le Cloud.

Bien que le Cloud computing et le Fog computing fournissent plus de stockage et de traitement, les systèmes de gestion de bases de données relationnelles restent incapables de fournir des services aux utilisateurs. Par conséquent, l'indexation des données est très importante. Plusieurs techniques d'indexation ont été proposées pour une indexation et une recherche de similarité efficace dans les réseaux de l'IdO dans l'espace métrique telle que L'arbre BCCF (arbre binaire basé sur les conteneurs du Cloud-Fog computing). Cette

structure est l'une des structures d'indexation dans l'espace métrique, cependant, cette technique souffre de certains inconvénients, tels que la complexité de l'algorithme de clustering k-means qui rend le processus de construction de l'arbre coûteux et le risque de chevauchement des arbres qui conduisent à un nombre trop élevé de nœuds visités lors de l'algorithme de recherche.

Objectifs

Dans ce projet, nous visons à proposer une nouvelle structure d'indexation de données massives de l'IdO plus optimale que l'arbre-BCCF basée sur le paradigme informatique Cloud-Fog-Edge computing pour diminuer le coût de la construction de l'index et réduit en tant que possible l'espace de recherche pour améliorer la qualité du processus de découverte et de récupération de données de l'IdO et de minimiser le nombre d'accès aux données.

Organisation du mémoire

Ce mémoire est organisé en cinq chapitres :

1. Chapitre 01 : **Notions mathématiques**
   Ce chapitre représente un état de l'art sur quelques notions mathématiques.
2. Chapitre 02 : **L'Internet des Objets et ses paradigmes émergents**
   Ce chapitre représente un état de l'art sur IdO, Cloud, Fog et Edge computing.
3. Chapitre 03 : **Techniques d'Indexation**
   Dans cette partie, nous présentons les différentes techniques d'indexation existantes dans la littérature dans l'espace métrique.
4. Chapitre 04 : **Conception**
   Dans ce chapitre, nous allons présenter notre contribution, à savoir la conception d'une nouvelle technique d'indexation pour optimiser la structure du BCCF. La méthode proposée est basée sur le prétraitement des données au niveau des couches Edge et Fog computing.
5. Chapitre 05 : **Implémentation et résultats de conception**
   Dans ce dernier, nous avons validé notre proposition en faisant des expérimentations sur différentes collections de données réelles.

Nous concluons ce mémoire par une conclusion générale et quelques perspectives sur le domaine.

Chapitre 1

Notions mathématiques

1.1 Introduction

Ce chapitre présente quelques notions mathématiques dont nous nous sommes prévalues dans cette étude. Ce chapitre est composé de trois sections : dans la première, nous définissons l'espace métrique et l'espace multidimensionnel. en suite, dans la deuxième, nous définissons la notion d'hyperplan et la notion de boule. Enfin, la fin du chapitre expose les requêtes par similarité avec leurs deux types.

1.2 Espace multi-dimensionnel

Un espace multidimensionnel est défini lorsque les éléments de l'ensemble sont considérés comme des vecteurs (c'est-à-dire que les données ont un nombre donné de dimensions), homogène ou hétérogène, dont les composants sont totalement ordonnées.

1.3 Espace métrique

En mathématiques, un espace métrique est un ensemble auquel on associe une notion de distance entre les éléments sous la forme d'une fonction également dénommée métrique.

Définition 1.1 (Métrique)
Soit O un ensemble d' éléments. Soit d une fonction de distance d'un couple d'éléments de O vers un réel positif ou nulle.

$$d : O * O \rightarrow \mathbb{R}^{+} \tag{1.1}$$

c'est-à-dire vérifiant les quatre axiomes suivants :

1. *Positivité (par définition) :*

$$\forall (x, y) \in O^2, d(x, y)\ 0; \tag{1.2}$$

2. *Identité :*

$$\forall x \in O, d(x, x) = 0; \tag{1.3}$$

3. *Symétrie :*

$$\forall (x, y) \in O^2, d(x, y) = d(y, x); \tag{1.4}$$

4. *Inégalité triangulaire :*

$$\forall (x, y, z) \in O^3, d(x, z)\ d(x, y) + d(y, z). \tag{1.5}$$

On note (O, d) est un espace métrique.

1.4 Fonction distance

Une fonction métrique ou de distance est une fonction $d(x, y)$ qui définit la distance entre les éléments d'un ensemble. Elle est généralement utilisée pour calculer la similarité entre des points de données. Dans cette étude, l'objectif est de réduire le nombre d'opérations effectuées pour répondre à une requête donnée.

À titre d'illustration, fournissons un exemple emblématique de métrique, la famille des distances de MINKOWSKI.

Distances de MINKOWSKI

Les distances de MINKOWSKI forment une famille de fonctions métriques, appelées « métriques L_p », qui peuvent être appliquées sur différents types de données à partir du moment où l'on peut les considérer, voire transformer, en vecteurs de nombres.

Définition 1.2 (Métriques L_p)
Soit $(x_1, ..., x_n) \in \mathbb{R}^n$ *et* $(y_1, ..., y_n) \in \mathbb{R}^n$ *deux vecteurs. Soit* $p \in \mathbb{R}$ *avec* $p \leq 1$

Alors, on définit :

$$L_p((x_1.....x_n), (y_1.....y_n)) = \sqrt[p]{\sum_{\forall i} |x_i - y_i|^p} \tag{1.6}$$

Quelques valeurs particulières du paramètre p sont plus souvent utilisées. Il s'agit des métriques :

- L_1 , communément appelée distance de Manhattan, qui se récrit plus simplement :

$$L_1((x_1, ..., x_n), (y_1, ..., y_n)) = \sum_{\forall i} |x_i - y_i|; \tag{1.7}$$

- L_2 , qui n'est autre que la distance euclidienne :

$$L_2((x_1.....x_n), (y_1.....y_n)) = \sqrt[2]{\sum_{\forall i} |x_i - y_i|^2}; \tag{1.8}$$

- L_∞, ou distance de CHEBYSHEV, communément appelée distance de l'échiquier, se ramène à :

$$L_\infty((x_1.....x_n), (y_1.....y_n)) = \max_{\forall i} |x_i - y_i|; \tag{1.9}$$

Une application pratique des métriques L_2 est le calcul de distance entre les vecteurs de différentes données. Nous en ferons usage au chapitre(4,5).

1.5 Notions de boule et d'hyper-plan

La géométrie classique peut être bousculée par l'emploi d'autres métriques. De manière générale, la fonction utilisée pour évaluer la distance entre éléments détermine une géométrie, ou topologie, de l'espace. Dans les espaces métriques, les auteurs exploitent souvent deux concepts clés :

- La boule.
- L'hyper-plan.

Nous l'utilisons également dans nos propositions. Présentons-les dans suivant :

1.5.1 Boule

Une boule est une notion topologique qui généralise celle de disque dans le plan euclidien et de sphère dans l'espace

Définition 1.3 (la boule)
Soit $(\mathcal{O}, d)$ un espace métrique. Soit $p \in \mathcal{O}$ un objet pivot et $r \in \mathbb{R}^+$. $B(\mathcal{O}, d, p, r)$ (ou seulement $B(p, r)$ lorsqu'il n'y pas d'ambiguïté sur l'espace métrique) est défini comme une boule qui partitionne l'espace en deux parties telle que :

$$\begin{aligned} I(\mathcal{O}, d, c, r) &= \{o \in \mathcal{O} : d(c, o) \leq r\}; \\ E(\mathcal{O}, d, c, r) &= \{o \in \mathcal{O} : d(c, o) > r\}; \end{aligned} \tag{1.10}$$

Définition 1.4 (la boule)
On appelle M une boule de centre c et de rayon r l'ensemble de tous les points de l'espace O qui sont situés à une distance du point c inférieure ou égale à R.

La figure 1.1 illustre la définition dans le cas euclidien.

les positions de deux boules dans un espace métrique Soit$(\mathcal{O}, d)$ un espace métrique. $c_1, c_2 \in \mathcal{O}$ et $.r_1, r_2 \in \mathbb{R}^+$ .Soit $S_1(\mathcal{O}, d, r_1, c_1)$ une boule avec c_1 centre et r_1 rayon. Soit $S_2(\mathcal{O}, d, r_2, c_2)$ une boule avec c_2 centre et r_2 rayon.

En distingue trois cas :

1. Intersection entre deux boules (Figure 1.2) :

   Il existe une intersection non vide entre S_1 et S_2 si, et seulement si :

$$d(o_1, o_2) \leq r_1 + r_2; \tag{1.11}$$

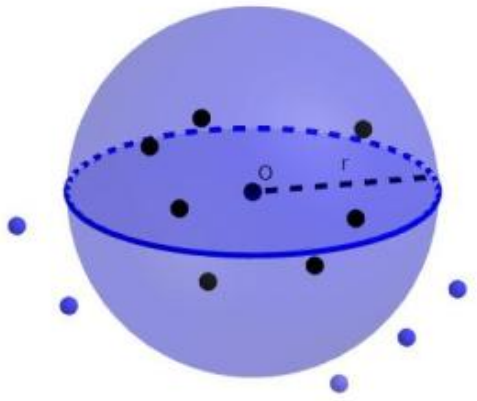


FIG. 1.1 : Visualisation de la structure d'une boule

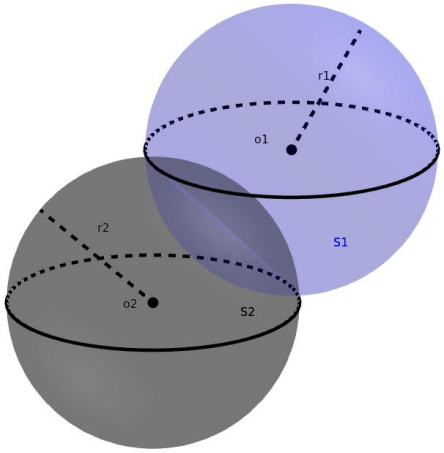


FIG. 1.2 : Visualisation de l'intersection entre deux boules

2. Pas d'intersection :

   Une intersection vide entre S_1 et S_2 si, et seulement si :

$$d(o_1, o_2) > r_1 + r_2; \tag{1.12}$$

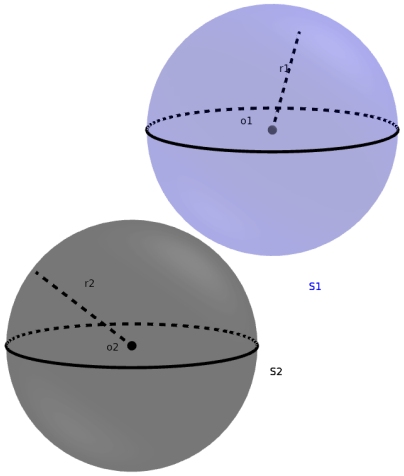


FIG. 1.3 : Visualisation de la non-interaction

3. Inclusion entre deux boules :

   La boule S_1 est incluse dans la boule S_2 si, et seulement si :

$$d(o_1, o_2) < r_1 - r_2 \tag{1.13}$$

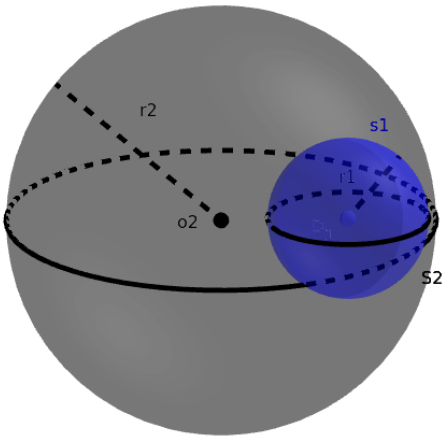


FIG. 1.4 : Visualisation graphique de l'inclusion entre deux boules

ou

$$d(o_1, o_2) = 0; r_1 < r_2 \tag{1.14}$$

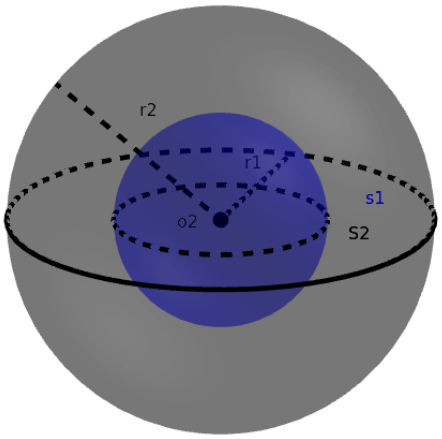


FIG. 1.5 : Visualisation graphique d'une inclusion entre deux boules

1.5.2 Hyper-plan

Un deuxième concept important dans un espace métrique est celui d'hyper-plan. Un hyper-plan est indirectement déterminé par deux éléments distincts,La figure 1.6 illustre ce concept.

Définition 1.5 (Hyper-plan généralisé)
Soit $(\mathcal{O}, d)$*un espace métrique. Soit* $(p_1, p_2) \in \mathcal{O}^2$ *deux pivots, avec* $(p_1, p_2) > 0$*. Alors* $H(\mathcal{O}, d, p_1, p_2)$ *– ou seulement*$H(p_1, p_2)$ *– définit un hyper-plan :*

$$H(\mathcal{O}, d, p_1, p_2) = o\ \mathcal{O} : d(p_1, o) = d(p_2, o); \tag{1.15}$$

qui permet de partitionner l'espace en deux sous-espaces :

$$\begin{aligned} G(\mathcal{O}, d, p_1, p_2) &= \{o\ \mathcal{O} : d(p_1, o) \leq d(p_2, o)\}; \\ D(\mathcal{O}, d, p_1, p_2) &= \{o\ \mathcal{O} : d(p_1, o) > d(p_2, o)\}; \end{aligned} \tag{1.16}$$

soit respectivement le demi-plan « gauche » au sens large –$G(p_1, p_2)$ *– et le demi-plan « droit » au sens strict –* $D(p_1, p_2)$*.*

C'est-à-dire que les objets $\in G(p_1, p_2)$ sont plus similaires de p_1 que de p_2 (ou à égale distance des deux), et les objets $\in D(p_1, p_2)$ sont plus similaires de p_2 que de p_1

1.6 Requête par similarité

Une requête de similarité est définie par un objet de requête et une contrainte sur la proximité requise pour qu'un objet figure dans l'ensemble de résultats. La réponse à une requête renvoie tous les objets qui satisfont aux conditions de sélection [5, 6]

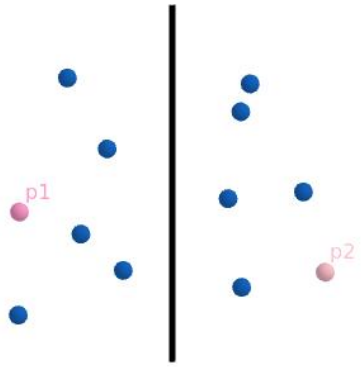


FIG. 1.6 : Visualisation graphique de la structure d'un plan

1.6.1 Requête par intervalle

Une requête par intervalle (q, r) est le type le plus courant de recherche, ce type de requête est spécifiée par objet q et un rayon r tell qu'il retrouvait tous les objets situés à une distance r de l'objet q, formellement donne par :

Définition 1.6 (Requête par intervalle)
Soit $(\mathcal{O}, d)$ un espace métrique. Soit $q \in E$ un point requête. Soit $r \in \mathbb{R}^+$ la distance maximale autorisée à l'objet q. Alors$(\mathcal{O}, d, q, r)$ définit une requête d'intervalle, dont la valeur est :

$$R(\mathcal{O}, d, q, r) = \{e \in E : d(q, e) \leq r\}. \tag{1.17}$$

La Figure 1.7 illustre ce concept.

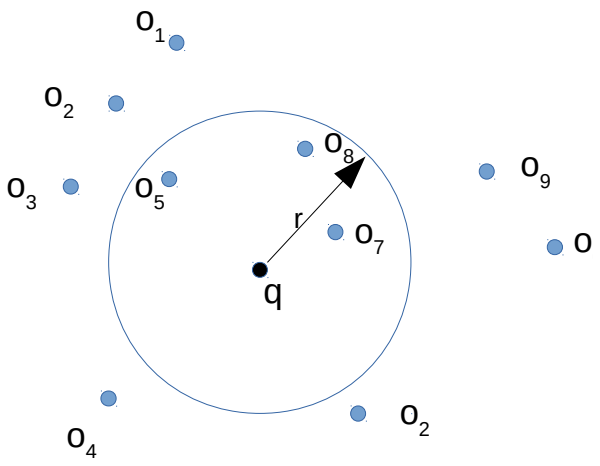


FIG. 1.7 : Exemple d'une requête intervalle $R(q, r)$

1.6.2 Plus proches voisins

Lorsque nous voulons rechercher des objets similaires à l'aide d'une recherche par intervalle, nous devons spécifier une distance maximale (rayon r) pour sélectionner les objets qualifiés. Mais il peut être difficile de spécifier le rayon sans une certaine connaissance des données et de la fonction de distance. car si le rayon d'interrogation spécifié est trop petit, la requête peut être renvoyée un ensemble vide. D'autre part, si le rayon de l'interrogation est trop grand, l'interrogation peut être coûteuse en termes de calcul et les ensembles de réponses contiennent de nombreux objets non similaires.

Une autre façon de rechercher des objets similaires consiste à utiliser la requête du plus proche voisin (q, k), nous recherchons les k objets les plus similaires à l'élément q, le nombre k est défini au préalable par l'utilisateur, ce type de requêtes peut être formalisé par la définition suivante.

Définition 1.7
Soit $(\mathcal{O}, d)$ *un espace métrique. Soit* $q \in E$ *un élément requête. Soit* $k \in N$ *le nombre recherché de réponses. Alors* $NN(\mathcal{O}, d, q, k)$ *définit une requête kNN, dont la valeur est* $S \sqsubseteq E$ *de telle sorte que* $|S| = k (sauf si |E| < k)$ *et* $\forall (s, e) \in S \times E, d(q, s) \leq d(q, e)$.

La Figure 1.8 illustre ce concept.

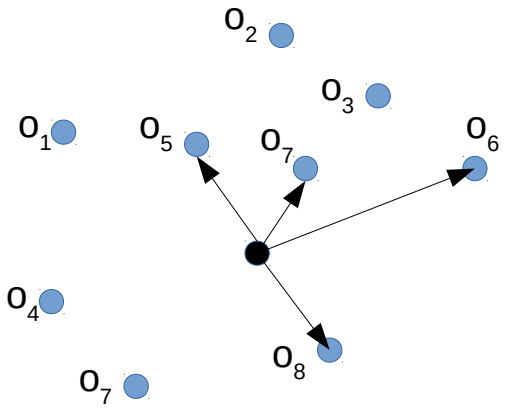


Fig. 1.8 : Exemple d'une requête plus proches voisins $R(q, r)$

1.7 Conclusion

Dans ce chapitre, nous avons présenté quelques généralités mathématiques sur les espaces métriques et les différentes métriques que nous pouvons utiliser pour les représentations de données. Dans le chapitre suivant, nous décrivons le concept de l'Internet des objets et ses paradigmes, à savoir, Cloud, Fog, Edge computing.

Chapitre 2

L'Internet des Objets et ses paradigmes émergents

2.1 Introduction

Avec le développement rapide des technologies de l'information et de la communication, nous nous acheminons vers une phase potentiellement la plus déroutante de la révolution Internet, à savoir l'Internet des objets" ou IdO" (en anglais (the Internet of Things ou IOT). L'IdO est un nouveau concept dans lequel différents objets sont connectés et communiquent entre eux via un réseau intelligent, générant une énorme quantité de données. Cette augmentation a entraîné l'émergence d'exigences strictes, notamment en matière de stockage et d'analyse des données, d'échange de données entre dispositifs, de sécurité et de confidentialité, ainsi que d'accès unifié et omniprésent.

Dans ce chapitre, nous présentons un état de l'art qui met en évidence certains détails techniques de notre cadre de travail. Ce chapitre se compose de trois sections. Dans la première section, nous présentons d'abord la technologie de l'IdO en détail. La deuxième partie présente un aperçu général du Cloud computing, ses types, ses modèles de livraison, ses avantages et ses limites. Enfin, nous consacrons le reste du chapitre à la définition et à la description des paradigmes du Fog computing et de l'Edge computing.

2.2 L'Internet des Objets (IdO)

2.2.1 Définition

Jusqu'à nos jours, il est difficile de trouver une définition claire et précise pour l'Internet des Objets (IdO). Nous avons donc rassemblé quelques définitions déjà existantes dans la littérature.

Définition 2.1
L'idée de base de ce concept est que la présence omniprésente autour de nous d'une variété de choses ou d'objets - comme les étiquettes d'identification par radiofréquence (RFID), les capteurs, les actionneurs, les téléphones portables, etc. - qui, grâce à des schémas d'adressage uniques, sont capables d'interagir les uns avec les autres et de collaborer avec leurs voisins pour atteindre des objectifs communs [7].

Définition 2.2
L'Internet des Objets (IdO) est une partie de futur Internet et peut être défini comme une infrastructure de réseau mondial dynamique avec capacité de configuration automatique basée sur des protocoles de communication standard et interopérables ou les objets ont des identités, des attributs physiques et des personnalités virtuelles et utilisent des interfaces intelligentes, et sont intégrés de façon transparente dans le réseau d'information. Ces objets peuvent interagir et communiquer entre eux et avec l'environnement en échangent des données et des informations sensibles à l'environnement tout en réagissant de manière autonome aux événements et les influençant en exécutant des processus qui déclenchent des actions et créent des services avec ou sans intervention humaine [8].

Définition 2.3
Selon l'Union internationale des télécommunications (ITU), l'IdO est l'infrastructure

mondiale de la société de l'information, qui fournit des services avancés en interconnectant des objets (physiques ou virtuels) à l'aide de la technologie Internet [2].

De notre point de vue, nous voyons que l'IdO forme un réseau d'objets (capteurs et actionneurs) interconnectés, qui peuvent communiquer entre eux et avec l'environnement via des protocoles de communication spécialisés pour échanger les données générées. Ces données sont stockées et analysées dans le Cloud ou le centre de données grâce à des techniques spécialisées de big data.

2.2.2 Le fonctionnement de l'IdO

Selon Perry Xiao [9], les quatre étapes du fonctionnement de l'IdO sont les suivantes :

- **Tout d'abord,** chaque « objet » connecté sur l'IdO doit avoir une identité unique. En raison de l'évolution des adresses IP (Internet Protocol) et de leurs générations, nous pouvons fournir des milliers d'adresses IP différentes, et nous devons avoir un identifiant unique qui peut être attribué à chaque objet physique sur terre.
- **Deuxièmement,** les « objets » doivent communiquer entre eux grâce aux nouvelles technologies sans fil existantes qui rendent la communication possible, telles que Wi-Fi, LoRaWAN, Bluetooth, Communication en champ proche (NFC), ZigBee, etc.
- **Troisièmement,** Chaque objet doit être équipé d'un capteur dans le but d'obtenir des informations sur l'environnement. Il existe de nombreux capteurs, notamment des capteurs de température, d'humidité, de lumière, de mouvement, de pression, des capteurs infrarouges, et de nouveaux capteurs tels que les ultrasons, qui deviennent de plus en plus petits, économiques et durables.
- **Quatrièmement,** grâce au microcontrôleur que les « objets » doivent avoir, on peut gérer les capteurs, les communications ainsi que l'exécution des tâches. Il existe de nombreux microcontrôleurs pouvant être utilisé dans IdO selon le besoin.
- **Enfin,** pour de pouvoir utiliser la puissance de calcul, le stockage des serveurs pour les calculs, l'analyse et l'affichage des données, il est recommandé d'utiliser les services cloud pour visualiser ce qui se passe et agir via une application pour téléphone mobile. Plusieurs grandes entreprises s'y intéressent déjà, comme "Watson", "IBM", "Google Cloud Platform", "Azure", "Microsoft", "Oracle Cloud", etc.

2.2.3 Les enjeux et les défis de l'IdO

L'augmentation rapide des objets connectés de l'IdO génère une quantité croissante de données. Cette augmentation entraîne de nombreux défis techniques [10], notamment :

- De nombreux nouveaux objets ou appareils intelligents sont automatiquement connectés au réseau. Cependant, l'IdO doit être capable de résoudre des problèmes

tels que les adresses IP, le traitement et la gestion des informations et des services. En outre, il doit prendre en charge les environnements à petite et grande échelle.

- Les objets de l'IdO doivent être programmés pour une configuration automatique afin de s'adapter à un environnement particulier sans intervention de l'utilisateur.

- Dans l'IdO, chaque objet intelligent a sa propre capacité de collecte d'informations, de traitement et de communication. Pour la communication et la coopération entre les objets intelligents de différents types, il est nécessaire de mettre en place une norme de communication commune capable de prendre en charge l'interopérabilité et l'hétérogénéité de ces dispositifs.

- En fonction du scénario et du contexte, les objets intelligents collectent soit une petite quantité de données, soit un énorme volume de données. Donc, en fonction de la quantité de données, le stockage doit être réaffecté.

- Dans l'IdO, le réseau est formé d'objets intelligents via Internet, et assurer la sécurité et la confidentialité est donc un grand défi. Dans l'IdO, parfois, un utilisateur empêche un autre utilisateur d'accéder à des informations particulières à un moment donné ou empêche une communication ou une transaction pour protéger les informations secrètes des concurrents. Le traitement de toutes ces situations est donc un grand défi.

2.2.4 Domaines d'applications de l'Ido

L'IdO trouvera ses applications dans presque tous les aspects de notre vie. Voici quelques exemples [11] :

Prévision des catastrophes naturelles : La combinaison de capteurs et leur coordination et simulation autonomes peuvent aider à prédire la prévalence des glissements de terrain ou diverses catastrophes naturelles et à prendre les mesures appropriées à l'avance.

Applications industrielles : L'IdO sera des applications industrielles, par exemple, la gestion des voitures pour les entreprises. L'Ido permet de surveiller leurs performances environnementales et de traiter les données pour identifier et sélectionner celles qui nécessitent une maintenance.

Maisons intelligentes : L'IdO peut aider à concevoir des maisons intelligentes grâce à l'automatisation des tâches quotidiennes telles que : la gestion de l'alimentation, l'interaction avec les appareils ménagers, la détection d'urgence, la sécurité, etc.

Applications médicales : L'IdO peut également être utilisé dans le secteur médical pour sauver des vies ou pour améliorer la qualité du suivi des patients par le suivi des paramètres de santé, suivi des activités, suivi des médicaments, etc.

Surveillance des pénuries d'eau : L'IdO permettra de détecter plus facilement les pénuries d'eau dans des endroits complètement différents. Les réseaux de capteurs associés

aux activités de simulation connexes peuvent non seulement surveiller les futures interventions liées à l'eau, telles que les bassins versants gestion, mais peut même être utilisés pour rappeler aux utilisateurs.

Application agricole : Un réseau de capteurs adaptés détectera les informations, les traitera et informera l'agriculteur à l'aide d'une infrastructure de communication, par exemple un message texte par téléphone portable sur la partie du champ qui nécessite un soin particulier.

La surveillance : L'IdO peut également être utilisé pour des applications de sécurité et de surveillance sur le terrain, tels que la surveillance de l'espace public ou privé, le suivi des personnes âgées, etc.

2.3 Cloud computing

Le terme "cloud computing" a fait couler beaucoup d'encre ces dernières années, cependant il n'existe pas encore de définition standard de ce terme. Pour les uns, le Cloud est un ensemble de services d'hébergement, tandis que pour d'autres, il s'agit de la fourniture d'une infrastructure à distance et pour d'autres encore, le Cloud est une fédération de services d'application à la demande. C'est pourquoi des travaux récents ont été consacrés à la normalisation et à la standardisation de la définition de ce paradigme. Par exemple, Vaquero et al. [12] ont examiné plus de 20 définitions différentes provenant de diverses sources pour déterminer une définition standard.

Dans ce travail, nous adopterons la définition du cloud computing de l'institut national des normes et de la technologie (NIST) [13], car elle couvre tous les aspects.

Définition 2.4
Cloud computing est un modèle permettant un accès réseau omniprésent, pratique et à la demande à un ensemble partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnées et libérées avec un minimum d'effort de gestion ou d'interaction avec le fournisseur de services. Ce modèle de Cloud computing se compose de cinq caractéristiques essentielles, de trois modèles de services et de quatre modèles de déploiement [13].

2.3.1 Les différents modèles de services de Cloud computing

Le Cloud computing est une technologie dans laquelle les ressources matérielles et logicielles telles que les applications spéciales, le processeur, le stockage et bien d'autres sont fournis aux utilisateurs en tant que service sur la base de la rémunération au fur et à mesure de leur utilisation et via Internet, car les besoins des utilisateurs ne sont pas les mêmes. Il existe plusieurs modèles de service pour le Cloud computing, selon la façon dont un service est fourni à l'utilisateur, le degré de contrôle que l'utilisateur a sur les ressources et le type de ressources que l'utilisateur a demandées. L'institut national des normes et de la technologie (NIST) définit trois modèles de services de base, à savoir IaaS, PaaS et SaaS comme le montre dans la Figure 2.1 [14, 15].

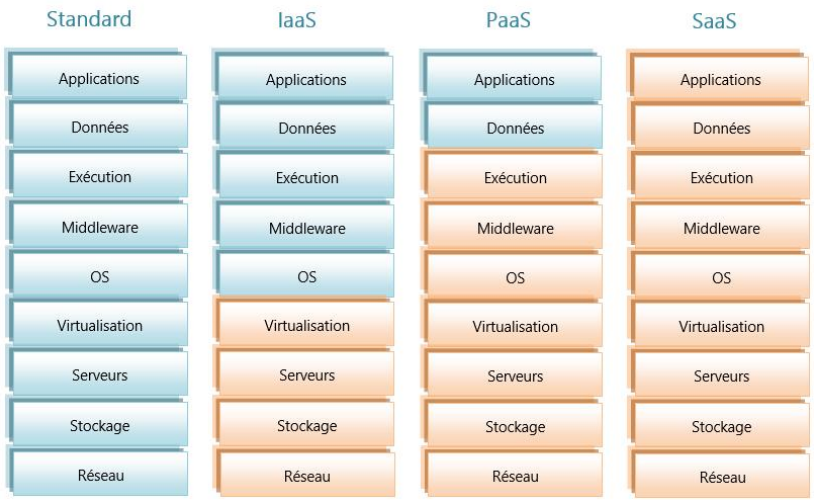


FIG. 2.1 : Les modèles de services de Cloud computing [16]

- **Software as a Service (SaaS) :** Le modèle SaaS se place sur le niveau supérieur selon la hiérarchie présentée dans la Figure 2.1. Le SaaS est considéré par beaucoup comme la nouvelle tendance en matière de fourniture de logiciels d'application. En outre, le SaaS peut être considéré comme une plate-forme en cloud multi-tenant [17]. Il partage des ressources communes et une instance d'une application et de la base de données sous-jacente pour prendre en charge plusieurs clients simultanément. Parmi les principaux exemples de fournisseurs de SaaS figurent SalesForce.com[1], NetSuite[2], et Microsoft (par exemple, Microsoft Office 365[3]).

- **Platform as a Service (PaaS) :** Il s'agit de la grande idée de fournir aux développeurs une plate-forme comprenant tous les systèmes et les environnements comprenant le cycle de vie de bout en bout du développement, des tests, du déploiement et de l'hébergement d'applications Web sophistiquées en tant que service fourni par un cloud. Il offre un moyen plus facile de développer des applications commerciales et divers services sur Internet. Deux exemples principaux de PaaS sont Google AppEngine[4] et Microsoft Azure[5].

- **Infrastructure as a Service (IaaS) :**

  L'IaaS est la fourniture de ressources aux utilisateurs en tant que service sur Internet par exemple le traitement, le stockage et la mise en réseau. Outre la plus grande flexibilité, l'un des principaux avantages de l'IaaS est le système de paiement basé sur l'utilisation. Cela permet aux clients de payer au fur et à mesure de leur croissance.

[1]SalesForce : https://www.salesforce.com/
[2]NetSuite : http://www.netsuite.com/
[3]Microsoft Office 365 : https://www.microsoft.com/en-us/microsoft-365/free-office-online-for-the-web
[4]Google AppEngine : https://console.cloud.google.com/projectselector/appengine
[5]Microsoft Azure : https://azure.microsoft.com/en-us/?b=16.26

Les principaux exemples sont Amazon EC2[6], GoGrid[7], et Mosso/Rackspace[8].

2.3.2 Les Cinq caractéristiques essentielles du Cloud computing

NIST a défini cinq caractéristiques principales. Ces caractéristiques sont brièvement expliquées comme suit [18] :

- **Libre-service à la demande :** Les clients ou les utilisateurs peuvent demander n'importe quelle ressource informatique à tout moment et la ressource est fournie automatiquement sans aucune interaction avec le fournisseur de service.
- **L'accès universel :** Les services fournis par le cloud sont accessibles via le réseau. L'accès à ces services est basé sur des mécanismes standards et peut être activé par différents appareils : mobile, tablette, ordinateur de bureau, ordinateur portable.
- **La mise en commun de ressources :** Les ressources mises à disposition sont communes à tous les clients et sont partagées dynamiquement entre eux en fonction de leurs besoins et l'emplacement exact des ressources concernées leur est inconnu.
- **Une élasticité rapide :** Les ressources sont mises à disposition et libérées à la demande, et à tout moment, le consommateur doit disposer exactement de la quantité de ressources dont il a besoin pour le produit. En fait, le consommateur doit être en mesure d'augmenter ou de diminuer la capacité de son système. En d'autres termes, elle définit la capacité d'une infrastructure donnée à s'adapter de manière dynamique avec les changements de demandes des utilisateurs. Cette caractéristique permet aux utilisateurs de provisionner rapidement de nouvelles ressources afin de pouvoir répondre à une augmentation ou une diminution de charge.
- **La mesure du service fourni :** Les ressources sont constamment surveillées et optimisées, et leur utilisation est gérée et communiquée aux clients.

2.3.3 Les modèles de déploiement du Cloud computing

Selon la définition du NIST, les services en nuage peuvent être déployés selon quatre modèles, à savoir le cloud privé, le cloud public, le cloud communautaire et le cloud hybride, qui correspondent à des usages différents [19] :

- **Cloud public :** sont le modèle typique de cloud computing, où les services cloud sont proposés par les fournisseurs de services cloud, tels qu'Amazon, IBM, Google, Microsoft, etc.
- **Cloud privé :** sont conçus pour être utilisés par une entité singulière et garantissent une grande confidentialité et configuration. Les clouds privés sont un bon choix pour les organisations qui ont besoin des infrastructures pour leurs applications.

[6]Amazon EC2 : https://aws.amazon.com/ec2/
[7]GoGrid : https://www.datapipe.com/gogrid/
[8]Mosso/Rackspace : https://www.rackspace.com/cloud

- **Cloud hybride :** sont simplement une combinaison des types de Cloud mentionnés ci-dessus. les clouds Hybride permettent aux utilisateurs d'avoir un contrôle plus fin sur l'infrastructure virtualisée et de combiner les capacités.

- **Cloud communautaire :** sont utilisés par une communauté d'utilisateurs et l'infrastructure est partagé entre plusieurs organisations. Un cloud communautaire se traduit par une décentralisation propriété du cloud par plusieurs organisations au sein de la communauté sans s'appuyant sur un grand fournisseur de cloud pour l'infrastructure informatique.

2.3.4 Avantages du Cloud computing

Le Cloud computing présente de nombreux avantages tels que :

- **Accessibilité facile :** on peut accéder a tous les services de Cloud computing à tout moment, sur tous les supports, via une connexion internet [20].

- **Disponibilité et fiabilité :** La raison pour laquelle l'environnement cloud est très fiable est que le fournisseur de services cloud utilise une sauvegarde complète, l'architecture modulaire du cloud et ses capacités d'équilibrage de charge. Tant que l'utilisateur est connecté à Internet, le service cloud est toujours disponible pour l'utilisateur et le client peuvent accéder à leurs données de n'importe où [18].

- **Capacité de stockage illimitée :** à l'aide de services Cloud, le client peut utiliser la capacité de stockage illimitée. Si les besoins de stockage du client augmentent, le client doit payer un peu plus pour utiliser une plus grande capacité de stockage fournie par le serveur Cloud. Ainsi, avec un faible coût d'installation, un utilisateur peut utiliser une capacité de stockage illimitée [18].

- **Facilité de mise à l'échelle :** les services Cloud peuvent facilement être mis à l'échelle selon les souhaits du consommateur. Par exemple, la capacité de stockage dans le cloud peut être augmentée jusqu'à To ou elle peut être aussi faible que quelques Go. [18].

- **Aucun coût de maintenance :** les fournisseurs de services Cloud n'ont pas besoin d'installer d'applications sur le PC, réduisant ainsi les coûts de maintenance. Le consommateur n'a qu'à payer pour le service qu'il a utilisé, et lorsque le serveur ou le lecteur de données s'use et tombe en panne, le fournisseur de services doit supporter le coût de remplacement [18].

2.3.5 Limites du Cloud computing

Le Cloud computing a beaucoup d'avantages. Pourtant, certaines entreprises n'ont pas intérêt à l'utilisation du Cloud, pour des raisons techniques et légales. Voici quelques inconvénients que présente le Cloud computing.

- **Les problèmes techniques :** Bien que les informations et les données stockées dans le cloud soient accessibles à tout moment et n'importe où, le système connaît parfois de graves défaillances. Les entreprises doivent être conscientes que cette technologie est toujours sujette à des défaillances et à d'autres problèmes techniques. Malgré les normes de maintenance élevées, même les meilleurs fournisseurs de services de cloud computing sont confrontés à ces problèmes [21].

- **Sécurité des données :** C'est le plus gros inconvénient du cloud, vous mettez vos données en ligne là où d'autres personnes peuvent potentiellement y accéder en cas de violation. C'est la principale raison pour laquelle de nombreuses entreprises hésitent à utiliser le cloud et malgré le succès du cloud pour conquérir le marché des petites et moyennes entreprises, il a peu de part dans les grandes entreprises. [18].

- **Dépendance du réseau :** Un autre inconvénient majeur du Cloud est sa dépendance à Internet, ou certain autre réseau local en cas de Cloud privé. Même si Internet est devenu omniprésent dans le monde développé, il trouve encore ses marques dans les pays en développement. Donc, utiliser le Cloud pour vos produits signifie essentiellement que vous ignorez cette partie de la population mondiale qui est sans Internet, et pour certains produits, ils peuvent être une population importante [18].

- **Contrôle limité :** Les consommateurs ont très peu de contrôle sur leurs produits dans le Cloud. Le plus grand contrôle dont ils disposent se trouve dans le modèle IaaS, où ils peuvent provisionner des machines virtuelles entières et les personnaliser en fonction de leurs besoins. Par contre, dans le modèle SaaS, ils ont le moins de contrôle, puisqu'ils peuvent seulement configurer certaines parties de l'application, mais ils n'ont aucun contrôle sur le reste [18].

- **Latence :** Les nouvelles applications dans le scénario IdO ont des exigences élevées en temps réel. Dans le modèle de cloud computing, les applications envoient des données au centre de données et obtiennent une réponse, ce qui augmente la latence du système. Par exemple, les véhicules de conduite autonome à grande vitesse nécessitent des millisecondes de temps de réponse. De graves conséquences se produiront une fois que la latence du système dépassera les attentes en raison de problèmes de réseau.

- **Bande passante :** La transmission de grandes quantités de données générées par l'IdO vers le cloud en temps réel entraînera une forte pression sur la bande passante du réseau. Par exemple, Boeing 787 génèrent plus de 5 Go/s de données, mais la bande passante entre un avion et les satellites est insuffisante pour prendre en charge la transmission en temps réel [22].

2.4 Les paradigmes Post-cloud

Les défis mentionnés dans la Section 2.3.5 ont nécessité l'ajout de nouveaux paradigmes entre le Cloud computing et les capteurs ou l'IdO. Différents paradigmes ont émergé à cette fin. Dans notre travail, nous nous intéressons aux deux paradigmes, à savoir le Fog computing et l'Edge computing (la Figure 2.2).

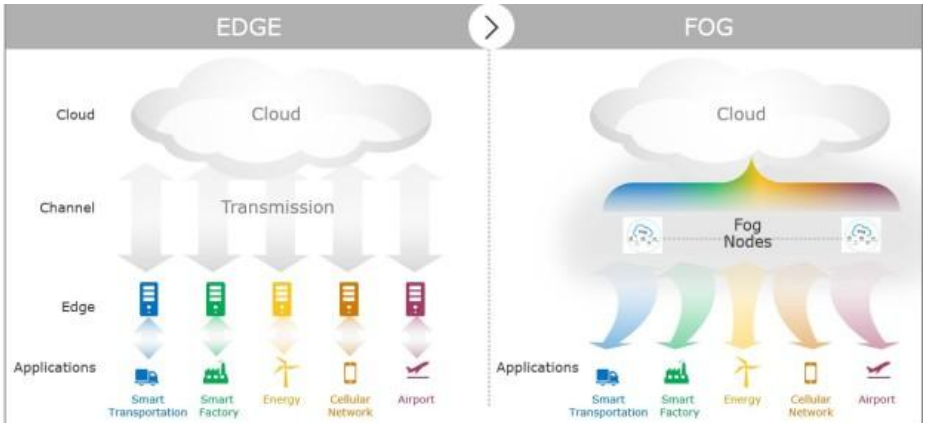


FIG. 2.2 : Fog vs Edge [23]

2.4.1 Fog computing

Le concept d'informatique de brouillard ou Fog (**F**rom c**O**re to ed**G**e) computing est une plate-forme hautement virtualisée qui fournit des services de calcul, de stockage et de mise en réseau entre les périphériques terminaux et les centres de données de cloud computing généralement, mais pas exclusivement, situés à la périphérie du réseau Figure 2.2 [3].

Ce paradigme a été introduit pour la première fois par Cisco en 2012 pour relever les défis des applications de l'IdO dans le Cloud computing.

Définition 2.5
Le Fog étend le Cloud pour être plus proche des objets qui produisent et agissent sur les données de l'IdO (Figure 2.2). Ces dispositifs, appelés nœuds du Fog, peuvent être déployés n'importe où avec une connexion réseau : sur le sol d'une usine, au sommet d'un poteau électrique, le long d'une voie ferrée, dans un véhicule ou sur une plate-forme pétrolière. Tout appareil doté d'une connectivité informatique, de stockage et de réseau peut être un nœud du Fog. Les exemples incluent les contrôleurs industriels, les commutateurs, les routeurs et les serveurs intégrés. ajoutez la référence

Yi et al [24] ont fourni une définition générale du fog computing. Il est indiqué comme :

Définition 2.6
Fog computing est une architecture informatique distribuée géographiquement avec un pool de ressources qui se compose d'un ou plusieurs appareils hétérogènes connectés de manière ubiquitaire (y compris les appareils périphériques) à la périphérie du réseau et non exclusivement soutenus de manière transparente par des services Cloud, pour fournir de manière collaborative un calcul, un stockage et une communication élastiques (et de nombreux autres nouveaux services et tâches) dans des environnements isolés à une large échelle de clients à proximité [24].

Architecture du Fog computing

La Figure 2.3 illustre une architecture à trois niveaux [25] du Fog computing, l'une des architectures de base et largement utilisées dans la littérature.

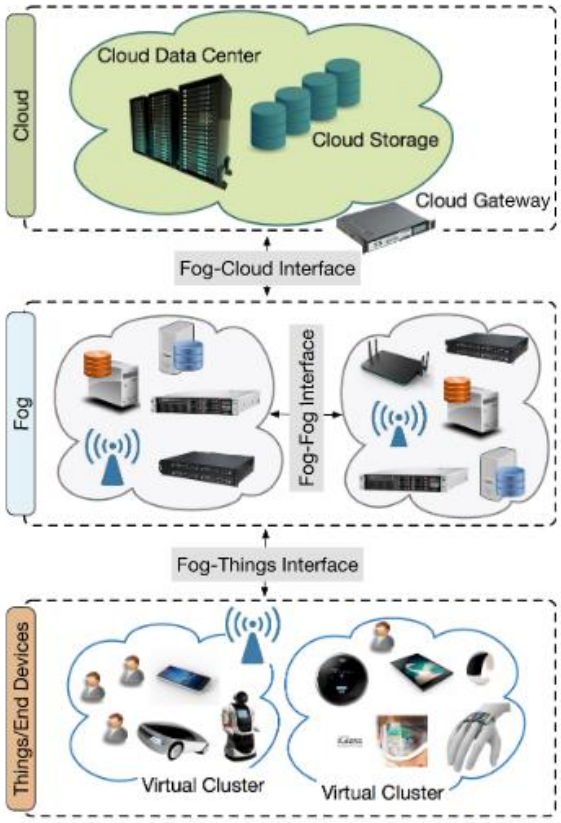


FIG. 2.3 : L'architecture à trois niveaux du Fog computing [26]

- **La couche Objets/Périphériques terminaux :** ce niveau comprend les périphériques de l'IdO, y compris les capteurs, les appareils portables tels que les smartphones, les tablettes, les montres intelligentes et autres. Ces périphériques sont souvent appelés nœuds terminaux.
- **La couche Fog computing :** les nœuds du Fog de cette couche sont constitués de périphériques réseaux tels qu'un routeur, une passerelle, un commutateur et des points d'accès. Ces nœuds du Fog peuvent partager en collaboration des installations de stockage et de calcul.
- **La couche Cloud computing :** les serveurs cloud résident dans le niveau le plus élevé. Ce niveau dispose de suffisamment de ressources de stockage et de calcul.

Comme le montre dans la Figure 2.3, il existe trois interfaces : (1) une interface **Fog-Cloud** devrait fournir des services de bout en bout, y compris la façon dont le nuage distribuera le service au Fog. (2) L'interface **Fog-Fog** permet à plusieurs nœuds ou systèmes de collaborer pour partager le stockage de données et les tâches de calcul. De

plus, (3) l'interface **Fog-Things** permettra en toute sécurité une utilisation efficace des ressources [27].

Caractérisation du Fog computing

Selon [28] et [24], les caractéristiques du Fog computing peuvent être résumées comme suit :

- **Détection de l'emplacement et faible latence :** Le Fog Computing prend en charge la connaissance de l'emplacement et peut déployer des nœuds de reconnaissance à différents endroits. De plus, comme Fog est plus proche du terminal, il permet de réduire la latence lors du traitement des données du terminal.
- **Distribution géographique :** Contrairement aux clouds centralisés, les services et applications fournis par Fog sont distribués et peuvent être déployés n'importe où.
- **Évolutivité :** Il existe de vastes réseaux de capteurs qui surveillent l'environnement environnant, et le Fog computing fournit des ressources de calcul et de stockage distribuées qui peuvent fonctionner avec ces périphériques à grande échelle.
- **Interactions en temps réel :** Les applications de calcul du Fog fournissent des interactions en temps réel entre les nœuds du Fog plutôt que le traitement par lots utilisé dans le Cloud.
- **Hétérogénéité :** Les nœuds du Fog , dans lesquels les terminaux sont conçus par différents fabricants, ont donc des formes différentes, et doivent être déployés en fonction de leurs plateformes. Fog peut fonctionner sur différentes plates-formes.
- **Interopérabilité :** Les composants du Fog peut interagir et travailler avec différents fournisseurs de services dans différents domaines.
- **Prise en charge de l'analyse en ligne et de l'interaction avec le cloud :** Le Fog est placé entre le cloud et les terminaux IoT et joue un rôle important dans l'absorption et le traitement des données à proximité des terminaux.
- **Accompagnement à la mobilité :** Un aspect important des applications du Fog est la capacité de communiquer directement avec les appareils mobiles et donc d'activer des méthodes de navigation, telles que le LISP[9] qui nécessite un système d'annuaire distribué.

Bénéfices du Fog computing

Le Fog computing étend le modèle de Cloud computing à la périphérie du réseau. Bien que le Fog et le Cloud utilisent des ressources similaires (réseau, stockage, traitement, etc.), et il partage bon nombre des mêmes mécanismes et fonctionnalités (virtualisation, multi-tenant) [29]. Le Fog computing apporte de nombreux bénéfices pour les objets de l'IdO. Ces bénéfices peuvent être résumés comme suit :

[9]LISP : Locator/Identifier Separation Protocol

- **Une plus grande flexibilité :** Avec les bons outils, les applications du Fog computing peuvent être rapidement développées et déployées. De plus, ces applications peuvent programmer la machine pour qu'elle fonctionne selon les besoins du client [30].

- **Faible latence :** Le Fog a la capacité de prendre en charge des services en temps réel (par exemple, jeux, streaming vidéo) [31].

- **Distribution géographique à grande échelle :** Le Fog computing peut fournir des ressources de calcul et de stockage distribuées à des applications de grande taille et largement distribuées [31].

- **Coûts d'exploitation réduits :** Économie de la bande passante du réseau en traitant les données sélectionnées localement au lieu de les envoyer vers le Cloud pour les analyser [30], en particulier, les données sensibles au temps des systèmes en temps réel.

- **Flexibilité et hétérogénéité :** Le Fog computing permet la collaboration de différents environnements physiques et infrastructures entre plusieurs services [32].

- **Évolutivité :** La proximité du Fog computing avec les terminaux permet de faire évoluer le nombre d'appareils et de services connectés [31].

Comparaison entre le Cloud et le Fog computing

Le concept de Fog computing est très similaire au Cloud computing, mais les quelques paramètres suivants montrent la différence entre ces deux concepts proches. Le tableau 2.1 résume la comparaison entre le Cloud et le Fog computing [33].

Les critères	**Cloud computing**	**Fog computing**
Latence	Élevée	Basse
Localisation du service	Dans l'Internet	Aux frontières du réseau
Distance Client-Serveur	Plusieurs sauts	Un seul saut
Gigue de retard	Élevée	Très Basse
Sécurité	Moins sécurisé	Plus sécurisé
Vulnérabilité	Haute probabilité	Très faible probabilité
Nombre de nœuds	Peu	Beaucoup
Support de mobilité	Limité	Supporté
Interaction en temps réel	Limité	Supportée
Géo-distribution	Centralisée	Distribuée
Mobilité	Prise en charge limitée	Prise en charge

Tab. 2.1 : Comparaison entre le Cloud et le Fog computing

2.4.2 Edge computing (informatique de périphérie)

L'Edge computing est une philosophie de mise en réseau visant à rapprocher les ressources informatiques aussi près que possible de la source de données (c.-à-d. les objets) afin de réduire la latence et l'utilisation de la bande passante. En termes plus simples, l'Edge computing signifie exécuter moins de processus dans le cloud et déplacer ces processus vers des emplacements locaux, tels que l'ordinateur d'un utilisateur, un appareil IdO ou un serveur de périphérie. Amener le calcul à la périphérie du réseau minimise la quantité de communication longue distance qui doit se produire entre un client et un serveur [34] notamment pour les applications en temps réel.

le Fog computing et le Edge computing déplacent tous deux le calcul et le stockage vers la périphérie du réseau et plus près des utilisateurs, mais ces paradigmes ne sont pas identiques. Le tableau 2.2 montre brièvement la différence entre ces deux modèles.

Les critères	**Fog computing**	**Edge computing**
Déploiement	Déployé dans les locaux des utilisateurs mobiles .	Déployé comme un centre de données traditionnel avec des capacités étendues.
Ressources	Appareil virtualisé avec stockage de données intégré, installation informatique et de communication.	Il utilise un serveur de périphérie similaire à un serveur de centre de données traditionnel.
Fonctionnement	Peut être adapté à partir de composants systèmes existants.	Il est entièrement construit comme un nouveau système ou un petit centre de données cloud.
Ressources consumer par rapport à cloud	La consommation d'énergie du brouillard est inférieure à celle des services cloud, mais la surcharge est élevée par rapport au cloud.	Edge utilise moins de ressources que la surcharge initiale du cloud à créer est élevée par rapport au cloud.
Contrôle du cloud	La consommation d'énergie du brouillard est inférieure à celle des services cloud, mais la surcharge est élevée par rapport au cloud.	Serveur Edge utilisant les tech-. nologies cloud et la virtualisation utilisée pour contrôler les composants Edge.
Contrôle d'opérateur de réseaux	Il peut ne pas être contrôlé par les opérateurs de réseau et il utilise une distribution personnalisée.	Permet aux opérateurs de réseaux mobiles d'améliorer les services existants avec Edge.

TAB. 2.2 : Fog computing **vs** Edge computing

Caractéristiques de l'Edge computing

L'Edge computing possède plusieurs caractéristiques similaires au Cloud computing. Cependant, les caractéristiques distinctives de l'Edge computing qui le rendent unique sont les suivantes :

Distribution géographique dense : L'Edge computing rapproche les services Cloud de l'utilisateur en déployant de nombreuses plateformes informatiques dans les réseaux de périphérie [35].

Aide à la mobilité : Comme le nombre d'appareils mobiles augmente rapidement, l'Edge computing prend également en charge la mobilité, comme le protocole LISP (Local ID Separation Protocol), pour communiquer directement avec les appareils mobiles [36].

Sensibilisation à l'emplacement : L'attribut de connaissance de l'emplacement de l'Edge permet aux utilisateurs mobiles d'accéder aux services du serveur Edge le plus proche de leur emplacement physique [36].

Proximité : Dans l'Edge computing, les ressources de calcul et services sont disponibles à proximité des utilisateurs qui peuvent améliorer leur expérience. La disponibilité des ressources de calcul et des services dans le voisinage local permet aux utilisateurs d'exploiter les informations de contexte de réseau pour prendre des décisions de déchargement et des décisions d'utilisation de service [36].

Faible latence : L'Edge computing rapprochent les ressources de calcul et les services des utilisateurs, ce qui réduit la latence d'accès aux services [36].

Sensibilisation au contexte : La connaissance du contexte est la caractéristique des appareils mobiles et peut être définie de manière interdépendante par rapport à la connaissance de l'emplacement. Les informations contextuelles de l'appareil mobile dans l'informatique Edge peuvent être utilisées pour prendre des décisions de déchargement et accéder aux services Edge [37].

Hétérogénéité : L'hétérogénéité dans l'Edge computing fait référence à l'existence de plates-formes, d'architectures, d'infrastructures, de technologies informatiques et de communication variées utilisées par les éléments du Edge computing (appareils finaux, serveurs Edge et réseaux)[36].

Bénéfices de l'Edge computing

Edge computing présente de nombreux avantages, comme indiqué ci-dessous [38] :

- **Vitesse et latence :** Limiter l'analyse des données à la Edge où elles sont générées élimine la latence, ce qui entraîne des temps de réponse plus rapides. Cela rend vos données plus pertinentes, utiles et exploitables. Edge computing réduit également les charges de trafic globales pour votre entreprise dans son ensemble, améliorant ainsi les performances de toutes vos applications et services d'entreprise.

- **Réduire les coûts :** Le coût sera réduit en raison des dépenses d'exploitation et de gestion des données plus faibles pour les appareils sur site par rapport aux centres de données et au Cloud.
- **améliorer l'efficacité énergétique :** Il permet une évolutivité et une efficacité énergétique améliorées pour les appareils IdO. Par exemple, les appareils domestiques intelligents comme Amazon Echo seront plus efficaces s'ils peuvent traiter les données près de la périphérie au lieu d'avoir à les renvoyer vers le Cloud.
- **Analyse des données en temps quasi réel :** L'analyse des données se fait en temps quasi réel puisqu'elle est analysée au niveau de l'appareil local et non sur un centre de données ou un cloud distant.
- **Réduire la charge du réseau :** Le calcul à la périphérie du réseau réduit la quantité de données qui circulent dans le réseau, car moins de données sont transmises des appareils locaux via un réseau vers un centre de données ou un cloud, réduisant ainsi les goulots d'étranglement du trafic réseau.

2.5 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art qui nous permet de comprendre certaines généralités sur l'Internet des objets, ainsi que sur les paradigmes du Cloud, du Fog, et de l'Edge computing.

Dans le chapitre suivant, nous décrivons le concept d'indexation des données à partir duquel nous présenterons les principales structures arborescentes proposées ces dernières années.

Chapitre 3

Technique d'indexation dans l'espace métrique

3.1 Introduction

L'IdO caractérise l'avenir dans lequel les appareils sont interconnectés via l'Internet et permettent une interaction homme-machine et machine-machine. En outre, la technologie L'IdO implique un grand groupe de dispositifs connectés pour capturer une énorme quantité de données. Le traitement et le stockage de cette énorme quantité de données nécessitent des solutions d'indexation évolutives et efficaces dans les trois niveaux Edge, Fog et Cloud computing pour les applications à grande échelle afin de permettre une recherche et une découverte rapides et efficaces de leurs données et services.

Dans ce chapitre, nous présentons un état de l'art sur les techniques d'indexation arborescente dans l'espace métrique existant dans la littérature, avec une petite étude sur les différences applications connexes des structures d'indexation.

3.2 Indexation arborescente dans l'espace métrique

L'indexation est une étape de l'organisation des données qui permet notamment leur accès efficace lors de l'exécution de requêtes de similarité. Par conséquent, un index a pour but de fournir un accès rapide aux objets d'une base de données, par la réduction de l'espace de recherche, du nombre de calculs de distance entre les objets et le temps de recherche.[39]. On peut classer les techniques d'indexation selon le type d'espace dans lequel elles sont créées. Deux catégories principales (i) Techniques d'indexation dans l'espace multidimensionnel (ii) techniques d'indexation dans l'espace métrique. Effectivement, les objets à indexer sont souvent plus complexes que de simples vecteurs (homogènes - ex. : espaces vectoriels-ou hétérogènes - ex. : tuples dans une base de données relationnelle). Tout ceci pose des questions sur les méthodes d'indexation et de recherche. En conséquence, le développement de l'indexation a été transféré des espaces multidimensionnels aux espaces métriques. Dans ce rapport, nous offrons un aperçu de certaines techniques d'indexation dans les espaces métriques, et aussi qui doivent respecter certaines contraintes, comme le montre la figure 3.1 pour être applicables à grande échelle.

- **Évolutivité** : La structure d'indexation doit être capable d'indexer la quantité toujours croissante de données sans perte de performance
- **Efficacité** : Contrairement aux structures traditionnelles d'accès physique aux entrées/sorties, elle cherche "seulement" à réduire le nombre d'entrées/sorties sur le disque. Un index de recherche de similarité doit également tenir compte du coût de calcul, car le calcul de la similarité entre deux objets peut être très élevé, voire plus coûteux que le nombre de lectures d'un disque optique.
- **Dynamique** : La structure de l'index doit être dynamique par rapport aux évolutions de la base de données, ce qui signifie qu'il n'est pas nécessaire de procéder à une réorganisation périodique coûteuse, mais d'assurer une indexation continue.
- **Indépendance des données** : La structure doit fournir de meilleures performances pour toute distribution possible des données, c'est-à-dire traiter efficace-

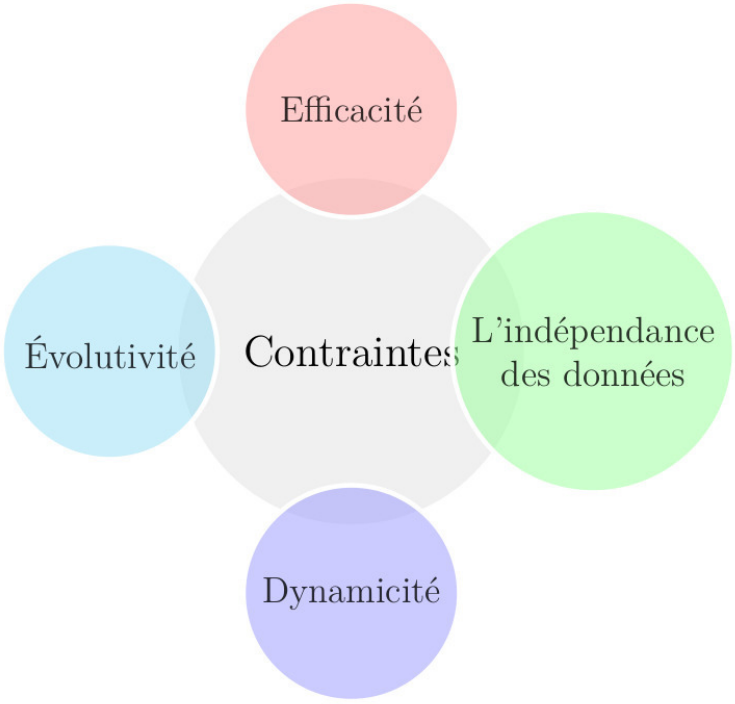


FIG. 3.1 : Exigences d'indexation des données massives [41]

ment les problèmes liés à la "malédiction des dimensions" ou à l'inefficacité dans les dimensions élevées.

Comme mentionné précédemment, les index dans l'espace multidimensionnel sont plus rigides en raison de leur forte dépendance au type ou, plus précisément, à leurs propriétés géométriques. Cette dépendance rend très difficile l'implémentation d'une structure flexible qui respecte les contraintes mentionnées dans la Figure 3.1 d'une part. Et, d'autre part, les techniques d'indexation et de recherche pour les espaces multidimensionnels ne fonctionnent généralement bien que pour les espaces à faible dimension. Même les méthodes d'indexation spécifiquement conçues pour des dimensions supérieures cessent généralement d'être efficaces lorsque la dimensionnalité dépasse vingt [40].

De ce fait, l'indexation s'est partiellement déplacée des espaces multidimensionnels vers les espaces métriques. L'espace métrique a été proposé comme une abstraction universelle pour le big data ou données massives. Parmi les principales caractéristiques de cet espace, il ne nécessite pas la structure intrinsèque des données, mais seulement une fonction de distance, avec les propriétés de non-négativité, de symétrie et d'inégalité triangulaire des paires de points de données qui sont mentionnées dans le chapitre section 1.3. L'avantage le plus important de l'indexation dans l'espace métrique est que tout type de données peut être indexé, car l'approche est basée uniquement sur le calcul des distances entre les objets et non sur leur contenu. Cela nous permet de créer une structure d'indexation générale (capable de traiter tout type de données).

Ce travail se concentre sur les espaces métriques par rapport aux espaces multidimensionnels. Par conséquent, nous avons réduit notre étude aux techniques d'indexation métrique (techniques basées sur les arbres). Deux approches principales de l'indexation

dans les espaces métriques sont présentées ci-dessous :

3.2.1 Non-partitionnement de l'espace(partitionnement de données)

L'idée principale du partitionnement des données est de créer des regroupements de données, également appelés " formes limites "[5].
L'arbre-M est une structure d'indexation équilibrée, incrémentale et métrique qui regroupe les données en boules en utilisant une fonction de distance. Elle est suffisamment performante en haute dimension. En revanche, elle souffre du problème du chevauchement.
L'arbre Slim [42] et leur extension l'arbre, est une version optimisée qui réorganise l'arbre M afin de réduire les problèmes de chevauchement , l'arbre Slim utilise le "fat-factor" qui fournit un moyen simple de quantifier le degré de chevauchement entre les nœuds d'un arbre métrique.
L'arbre-supre-M [43] une amélioration de l'arbre-M par utilisation des différentes fonctions de distance, la distance euclidienne, la distance Hausdor (sur les sous-ensembles), la distance Édit et la distance Dog-Keeper (sur les sous-séquences), pour calculer la distance entre les objets dans les nœuds internes et ses sous-arbres, l'arbre-supre-M capable de répondre à la demande de sous-séquences ou de sous-ensembles telle que la recherche pour une séquence partielle similaire d'une scène dans un film, ou un objet similaire dans une image.
Murgante et al [44] ont proposé une nouvelle structure d'indexation métrique appelée L'arbre MX basée sur l'original Structure (l'arbre-M).L'arbre-MX adapte le concept de super nœuds inspiré de la structure l'arbre x [45] qui évite ainsi la division insatisfaisante des nœuds, réduit le coût de calcul et l'étend complètement à l'espace métrique où la complexité temporelle est réduite à $\mathcal{O}(n^2)$ sans régler aucun paramètre ,Contrairement à L'arbre-M, où la complexité temporelle attend pour $\mathcal{O}(n^3)$.
L'arbre PM [46] (Pivot based Metric Indexing) est proposé pour résoudre le problème de recherche dans l'arbre-M. L'arbre PM combine la cartographie pivot avec l'arbre-M pour éviter les calculs de distances inutiles. L'arbre DSC (Dynamic Set of Clusters) [47] une nouvelle structure métrique dynamique qui réduit la consommation de mémoire, combine les deux structures(DSAT, LC) afin de réduire le nombre de calculs de distances.
À cause des données manquantes générées par différents domaines d'application, les structures d'indexation sont déformées, où cette dernière produit une fausse réponse à la requête. Pour résoudre ce problème, Brinis et al [48] proposent la structure arbre Hollow, qui permet de gérer les données manquantes sans en altérer la structure. Hollow tree est une nouvelle méthode d'accès métrique qui utilise la technique CFMLI (Complete First and Missing Last Insert) pour fournir une stratégie de construction d'indices métriques. Cette stratégie consiste à indexer toutes les données complètes dans un premier temps pour créer une structure cohérente puis à insérer les éléments avec les valeurs manquantes au niveau des nœuds des feuilles. Tout ceci est réalisé par la technique ObAD (Observed Attribute Distance), qui permet de comparer les éléments avec des valeurs manquantes sur la base de fonctions de distance.

3.2.2 partitionnement de l'espace

Du fait du principal problème associé aux techniques d'indexation basées sur le partitionnement des données, à savoir les chevauchements entre les formes limites, il existe une autre approche où les intersections de régions sont inexistantes. Cette approche est basée sur le partitionnement de l'espace. Cette approche comporte deux sous-approches :

Le partitionnement par boules

Dans cette catégorie, le choix des pivots pour le découpage de l'espace joue un rôle très important pour la construction des index, Arbre -VP (Vantage Point) [49] se fonde sur la recherche de l'élément médian d'un ensemble d'objets. Arbre-mVP [50] est une généralisation de l'arbre VP (mVP multivoie), les noeuds sont divisés en quantiles. Ce principe de partitionnement résout le problème du chevauchement entre les formes d'inclusion.
L'arbre métrique à mémoire (MM-tree)[51] utilise le principe du partitionnement de boules, mais il est basé sur l'exploitation des régions obtenues à partir de l'intersection entre deux boules à la fois.
L'arbre-Onion carelo2011slicing est une extension de l'arbre MM, basée sur le découpage dynamique en mémoire de l'espace métrique pour fournir une indexation rapide de données complexes.
L'arbre-XM (eXtended Metric tree) [52] partitionne l'espace à l'aide de sphères en limitant le volume des régions extérieures des sphères ; en créant des régions étendues et en créant des régions étendues et en les insérant dans des listes liées nommées régions étendues, et aussi en excluant les ensembles vides qui ne contiennent pas d'objets. L'arbre -Boule [53] est un arbre binaire dans lequel chaque nœud définit une boule de dimension d, contenant un sous-ensemble des points à rechercher. Chaque point est affecté à l'une ou l'autre boule de la partition en fonction de sa distance au centre de la boule.
L'arbre-Ball*[54] est une amélioration de l'arbre-ball [55] où les distributions sont plus équilibrées et les requêtes de recherche de plus proches voisins plus efficaces.

Le partitionnement par hyper-plans

L'arbre-GH (Generalized Hyper-plane) [56] est une structure d'indexation binaire qui divise récursivement l'espace en deux sous-espaces par l'hyperplan qui est défini par les deux points représentatifs ou pivots (les deux points les plus éloignés comme dans [57] [58] et [59]) et le reste des points sont distribués en fonction de la distance entre ces pivots.
L'arbre-GNAT [60] est une généralisation de l'arbre-GH qui utilise m pivots dans chaque nœud interne au lieu de deux (c'est-à-dire que l'arbre-GNAT est un arbre m-aire).
L'arbre-GHB (Generalized Hyper-plane Bucketed) [61] est une amélioration de l'arbre GH. L'objectif de l'arborescence-GHB est de créer une structure d'indexation équilibrée avec un coût de construction moindre grâce à un nouveau type de nœud qu'ils ont appelé bucket. Ces types de nœuds se trouvent à la feuille qui a une capacité limitée pour stocker un sous-ensemble de données les plus similaires afin d'améliorer le processus de recherche.
L'arbre-CD (Clustering-based Dynamic indexing) Le principe de cette technique est la partition récursive de l'espace en deux régions. Deux pivots sont choisis à chaque fois, et chacun est associé aux objets les plus proches.

L'arbre-SPB (Space-filling curve and Pivot-based B +-tree) [62], cette méthode est proposée pour améliorer l'efficacité de la recherche de similarité, supporter un grand nombre d'objets complexes et réduire le coût en termes de stockage, de construction et de recherche.

Partitionnement Hybride

Certaines structures d'indexation utilisent les deux techniques à la fois, comme :
L'arbre-IM [63] est une structure proposée pour traiter le problème de dégénérescence de l'index posé par la quatrième région de l'arbre-MM et l'oignon. L'arbre-IM sélectionne les deux points les plus éloignés comme pivots et divise la quatrième région en deux en utilisant un hyperplan et une hyper-sphère.
L'arbre-NOBH [64](Non-Overlapping Balls and Hyper-planes tree) divise l'espace métrique en hyperplans et en hyper-sphères afin d'organiser les données en régions non chevauchantes et de réduire le nombre de distances de calcul nécessaires pour répondre aux requêtes.

Nous pouvons introduire une taxonomie simple, comme illustré dans le tableau ci-dessous qui représente plusieurs techniques d'indexation dans les espaces métriques.

3.3 L'arbre BCCF

3.3.1 Présentation

L'arbre BCCF [83] (Binary tree based on Containers at the Cloud-Fog computing level) est une nouvelle méthode récemment proposée par l'équipe GADM du laboratoire LABSTIC. Cette technique vise à indexer les données massives de l'IdO. Elle est basée sur le partitionnement des données via l'algorithme K-means qui permet de regrouper les objets les plus similaires. À chaque étape du processus de construction récursif, des pivots sont initiés par les centres de gravité retenus par l'algorithme k-means. Voir la Figure 3.2.

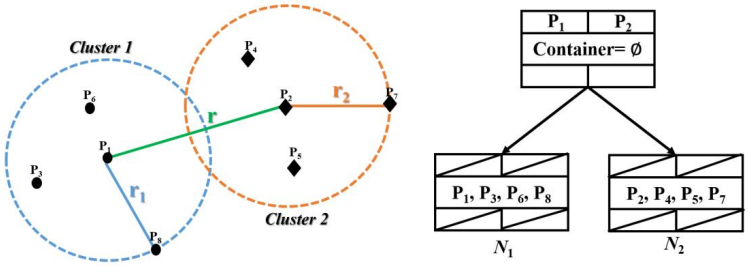


Fig. 3.2 : Partitionnement de espace avec l'arbre BCCF

La structure BCCF se compose de deux niveaux :

Approches			Applications connexes
Partitionnement de l'espace	boules	Arbre-VP	Reconnaissance des formes et traitement des images [65] Indexation et recherche d'images [66] Stockage des données de morphologie neuronale[67] Recherche de similarités sur le cloud computing [68] Détection de logiciels malveillants [69] Regroupement des images similaires [70]
		Arbre-mVP	Recherche d'images dans les systèmes de surveillance. [71]
		Arbre-MM	Récupération d'images. [63]
		L'arbre Onion	
		Arbre-XM	Recherche d'informations sur le Web [72].
		Arbre Boule	Reconnaissance d'esquisses de visages [53]. Classification en haute dimension [73].
		Arbre-Boule*	Regroupement et mise en correspondance pour la reconnaissance des classes d'objets [74].
	Hyper-plans	Arbre-GH	Recherche d'images par contenu [75]
		Arbre-GNAT	Indexation et recherche de similarité de données d'images de visages
		Arbre-EGNA	Accélérateur de requête de base de données
		Arbre-GHB	Indexation et recherche de similarité de données d'images de visages
		Arbre-CD	Indexation et recherche de similarité de données d'images de visages
		Arbre-SPB	Recherche multimédia, Reconnaissance des formes et biologie computationnelle
	Hybride	Arbre-IM	Reconnaissance des formes et biologie computationnelle. Accélérateur de requête de base de données
		Arbre-NOBH	Indexation et recherche de similarité de données d'images de visages.
Non partitionnement de l'espace	Arbre-M		Recherche de similarité dans une grande base de données multimédia [76] Recherche du voisin le plus proche [77] Accélérateur de requête de base de données [78] Système de recommandation [78] Classification[79]
	Arbre-Slim		Indexation de vidéo et recherche de similarité [80]
	Arbre-supre-M		Applications de recherche de similarités [43]
	Arbre-MX		Indexation et recherche de similarité
	Arbre-PM		Recherche de similarité dans les bases de données multimédias [81]
	Arbre-DSC		Recherche de similarité dans les bases de données multimédias [47]
	Arbre-Hollow		Stocker et récupérer de grands volumes de données [82]

Tab. 3.2 : Taxonomie des techniques d'indexation arborescentes dans les espaces métriques

1. **Niveau nœuds internes :** Les nœuds de ce niveau contiennent uniquement deux pivots (p_1, p_2) et deux pointeurs vers les sous arbres gauche et droit. Pour les conteneurs, tous les nœuds de ce niveau ont des conteneurs vides.

2. **Niveau nœuds feuilles :** Les nœuds de ce niveau ne contiennent pas des pivots ni des pointeurs, mais contiennent seulement des conteneurs qui stockent des ensembles d'objets dont la taille d'un conteneur est inférieure ou égale à C_{max}.

3.3.2 les points forts et les point faibles de la structure BCCF

1. **Points forts**

   - Structure hiérarchique.
   - Une séparation efficace des objets dans les nœuds de l'arbre.
   - Amélioration de l'efficacité de l'algorithme de recherche.

2. **Les lacunes**

   - La construction coûteuse de l'arbre a la cause d'utilisation de l'algorithme du K-means.
   - Chevauchement entre les sous-arbres .
   - Un grand nombre de nœuds visités lors de la récupération d'objets, qui influence négativement a efficacité d'algorithme de recherche.

3.4 conclusion

Dans ce chapitre, nous avons exposé l'état de l'art des techniques d'indexation. Plusieurs méthodes ont été discutées. En se basant sur les deux méthodes d'indexation, le partitionnement et le non-partitionnement de l'espace, nous pouvons introduire une légère taxonomie de quelques techniques d'indexation dans les espaces métriques. En conclusion :

- Nous avons noté, comme plusieurs auteurs l'ont déjà fait, que les espaces métriques sont devenus un modèle populaire pour contourner les limitations des espaces vectoriels dans diverses applications.
- La structure BCCF s'est avérée efficace, mais elle doit encore être optimisée pour diminuer le coût de la construction de l'index et de l'espace de recherche.

Chapitre 4

BCCF* Optimisation de BCCF

4.1 Introduction

Le principal objectif de notre étude est de garantir la performance de l'indexation, ainsi que la flexibilité et la rapidité de la recherche d'information sur les données Ido. Pour atteindre cet objectif, plusieurs structures d'indexation ont été proposées, notamment BCCF (Chapitre 2.5 Section 3.3, Cette dernière a démontré ces performances, grâce à l'utilisation de l'algorithme de k-means pour le partitionnement de données qui a conduit à une séparation efficace des objets dans les nœuds de l'arbre, ce qui améliore la qualité du processus de recherche.

Malheureusement, cette technique présente des désavantages mentionnés dans la sous-section 3.3.2 qui diminuent l'efficacité de cette structure d'indexation.

Pour résoudre ce problème et améliorer BCCF suggère de :

1. De proposer l'application de l'algorithme de regroupement a la totalité des données (au niveau du "Edge computing") avant l'indexation.

2. Utiliser la technique des hyperplans pour partitionner l'espace au lieu de l'algorithme des k-means pour éviter leur coût de calcul.

3. pour surmonter le problème des partitions spatiales qui se chevauchent, nous proposons un algorithme visant à estimer le taux de chevauchement entre les partitions spatiales avant de créer de nouveaux index. Cette approche permet de construire plusieurs arbres non chevauchants et interconnectés dans le but d'améliorer la qualité et l'efficacité des algorithmes de recherche.

Le présent chapitre est composé de plusieurs sections : Nous présentons dans un premier temps l'architecture du système proposé dans la section 4.2, Ensuite nous détaillons la méthode de regroupement dans la section 4.3 et la méthode d'estimation et de décision proposée dans la section **??**, Puis nous exposons la structure proposée (L'arbre BCCF*) dans la section 4.4. Enfin dans la section 4.5 nous détaillons un algorithme de recherche pour répondre aux requêtes de type K-NN.

4.2 L'architecture du système proposé

Le schéma 4.1 illustre l'architecture du système d'indexation proposé et présente les principaux composants et leurs tâches respectives.

Le scénario du système proposé se déroule dans la couche des objets connectés ou couche des capteurs Ido. Cette couche joue un rôle important dans l'indexation des données d'Ido : (1) la détection, (2) la collecte et (3) la communication. Cette couche est constituée de millions de capteurs connectés hétérogènes provenant de différents domaines (industriel, commercial, santé, villes intelligentes, transport, agriculteurs, etc.), ces capteurs génèrent de grandes quantités de données.

Le rôle de la couche Edge est : (1) la représentation, (2) le stockage temporaire, et (3) le pré-traitement, Les données reçues par la couche inférieure (capteurs IoT) sont des

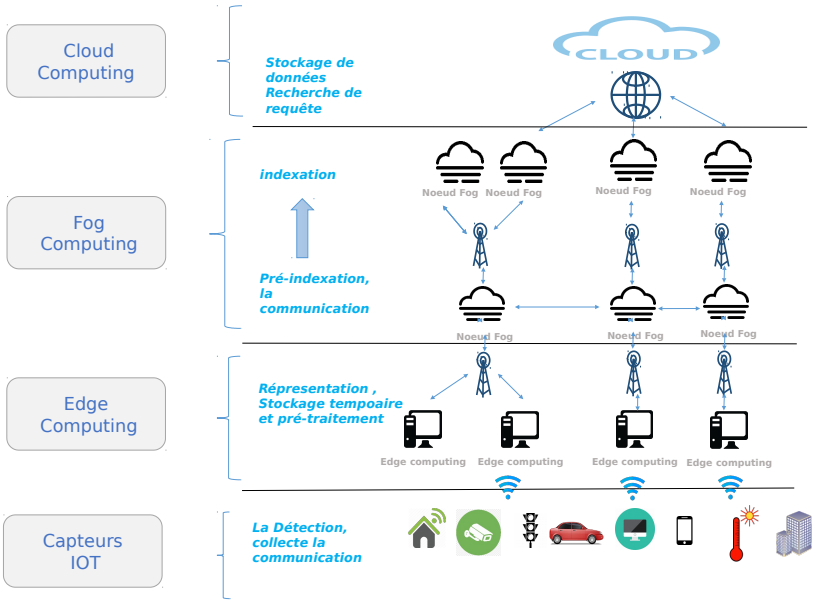


FIG. 4.1 : Architecture du système d'indexation

données de différents types (string, vecteur, etc.) Les nœuds Edge vont représenter ces données dans des formats lisibles par une machine Ensuite, les nœuds Edge traitent les données avant de les envoyer à la couche Fog pour l'indexation.

La couche de Fog se compose de deux niveaux : Le rôle du premier niveau est (1) la communication et (2) la pré-indexation des données pour assurer une meilleure indexation, le rôle du deuxième niveau est l'indexation, les nœuds du Fog indexeront ces données par la structure d'indexation proposée ($BCCF^*$), Notre structure arborescente est stockée dans cette couche pour développer un mécanisme d'indexation efficace.

La couche Cloud, son rôle est de : (a) la découverte, (b) la requête, également (c) le stockage et (d) l'analyse. Les données réelles indexées dans la couche du Fog seront stockées et traitées dans le nuage en raison de sa grande capacité de stockage et de traitement.

Dans le processus de recherche de similarité, les utilisateurs du système envoient leurs requêtes de recherche à la couche informatique du Fog, et lorsque la requête est reçue par le nœud de Fog de niveau inférieur, ce dernier envoie la requête aux nœuds de Fog correspondants, Chaque nœud de Fog exécute un algorithme de recherche de similarité parallèle sur tous les arbres candidats (BCCF*),

Lorsque les nœuds de Fog localisent l'ensemble des réponses dans la structure d'index locale, ils envoient des demandes au comptage en nuage pour récupérer les données réelles recherchées et les renvoyer au nœud de Fog, qui combine les réponses reçues, sélectionne la bonne réponse et l'envoie à l'utilisateur.

4.3 Pré-traitement : regroupement

Afin d'améliorer l'arbre BCCF, nous proposons de procéder à l'application d'un algorithme de regroupement à l'ensemble des données (au niveau de l'Edge-computing) avant l'indexation. Ce processus réduit le coût de l'algorithme de construction, cependant l'algorithme K-means nécessite de connaître le nombre de groupes dès le début, dans le BCCF, k= 2, car l'arbre est binaire, mais dans notre cas, le K est inconnu.

Dans notre perspective, l'algorithme DBSCAN [84] est le plus approprié. DBSCAN est un algorithme de regroupement basé sur la densité, conçu pour découvrir des clusters de formes arbitraires. L'idée principale de DBSCAN est la suivante : Il recherche, pour chaque point, les points qui font partie de son voisinage immédiat, et il regroupe tous les points qui peuvent être atteints de proche en proche. Une fois tous les points regroupés, les groupes de points sont considérés soit comme des clusters, soit comme des points aberrants, si le nombre d'observations est inférieur à le paramètre MinPoints.

Dans ce dernier cas, il n'est pas nécessaire de connaître à l'avance le nombre de clusters souhaité, alors que d'autres méthodes d'approprié, comme les k-means, nécessitent en entrée le nombre de clusters, qui n'est pas toujours facile à déterminer. La version de base de DBSCAN permet uniquement de regrouper des éléments similaires sans déterminer un représentant pour chaque groupe (centre, rayon). Cette information manquante est très importante dans notre proposition. Dans ce but, une nouvelle version de DBSCAN est proposée (Algorithme 1) pour prendre en compte les exigences mentionnées précédemment.

Algorithm 1 DBSCAN

Input O : set of data , eps node ,$Minpts$ node

Output C Clustre,C_c Centres des Clustres,R_c Rayons des clustres,

ClusterID=nextId(Noise) **for** $i \in O.size$ **do** Point=o.get(i)

if $Point.ClId = UNCLASSIFIED$ **then**

  if $ExpandCluster(O, Point, ClusterId, Eps, MinPts)$ **then**

    ClusterID = nextId(ClusterID)

  end if

end if

end for

for $i \in O.size$ **do**

  calcul C_{c_i}

  calcul R_{c_i}

end for

return C ,C_c ,R_c

4.4 L'approche d'indexation proposée

BCCF* est une nouvelle version destinée à optimiser l'arbre-BCCF ; Nous proposons donc de remplacer le coûteux algorithme k-means par une méthode de regroupement (dans la couche Edge), un algorithme permettant d'estimer le taux de chevauchement entre les partitions de l'espace (dans la couche Fog inférieure), avant la création de nouveaux index. Toutefois, l'index proposé est basé sur la division de l'espace de manière récursive, (dans la couche de Fog), en deux sous-espaces (ou partitions). Nous définissons les nœuds N d'un arbre-BCCF* comme suit :

1. Nœud feuille : se compose deux sous-ensembles d'objets indexés :où :

$$((p_1, p_2, r, r_1, r_2, c, sac_1, sac_2) \in \mathcal{O}^2 \times (R^+)^3 * \mathcal{O} * E^2 \tag{4.1}$$

   Tous les nœuds feuilles à ce niveau sont stockés dans le Cloud ;

2. Nœud interne : est un sextuplé

$$((p_1, p_2, r, c, , N_1, N_2) \in \mathcal{O}^2 \times (R^+)^3 * \mathcal{O} * E^2 \tag{4.2}$$

Les nœuds internes sont stockés dans le fog ;
Où :

- (c, r) Ils sont le centre(c) et le rayon(r) du groupe de tous les objets existé
- (p_1, p_2) sont deux objets distincts, c'est-à-dire avec $d(p_1, p_2) > 0$, appelés pivots, sont initiés par les deux objets les plus éloignés.
- sac_1 Ensemble de données la plus similaire a p_1 $|sac_1| <= c_{max}$
- sac_2 ensemble de données la plus similaire a p_2 $|sac_2| <= c_{max}$
- (r_1, r_2) sont les distances à l'objet le plus éloigné dans les sous arbre qui concerne $p_1 et p_2$ respectivement, c'est-à-dire r_i=max {d(p_i , o), $\forall o \in$sac$_i$ }
- N_1, N_2 sont deux sous-arbres, de telle sorte que :
  - $N_1 = o \in N : d(p_1, o) \leq r \wedge d(p_2, o) > r$ pour les données montées dans la boule partielle centrée sur p_1
  - $N_1 = o \in N : d(p_1, o) \leq r \wedge d(p_2, o) > r$ pour les données montée dans la boule partielle centrée p_1

La figure 4.2 illustre de façon informelle la façon dont l'arbre est conçu à partir d'un nœud.

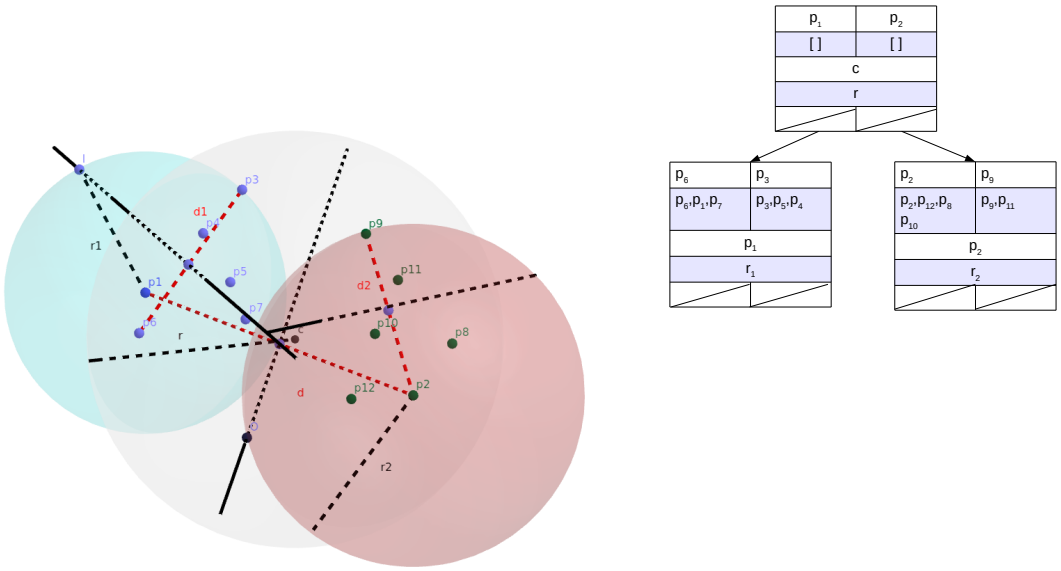


FIG. 4.2 : Partitionnement de l'espace avec un arbre BCCF*

La description formelle du processus de création de d'un nœud d'arbres est représentée dans l'algorithme 2

Algorithm 2 creation of node

Input S : set of data
Output N : node

1: (p_1, p_2) = choice of pivots (S) ;
2: c = center(S) ;
3: r = radius(S, c) ;
4: **for** v $\in S$ **do**
5: **if** $d(p_1, v) \leq d(p_2, v)$ **then**
6: $sac_1.append(v)$
7: **else**
8: $sac_2.append(v)$;
9: **end if**
10: **end for**
11: $N.c = c$
12: $N.r = r$
13: $N.sac_1 = sac_1$
14: $N.sac_2 = sac_2$
15: **return** N

4.4.1 Construction de l'index dans l'arbre BCCF*

Le processus incrémental de construction d'un arbre BCCF* consiste à insérer des objets de haut en bas. La description formelle du processus de construction de l'arbre BCCF* est donnée dans l'algorithme 3. Tout d'abord, le premier ensemble de données arrive. L'algorithme de recherche à deux axes de l'extérieur est utilisé pour tous les objets afin de diviser le conteneur en deux conteneurs de sorte que chaque élément du conteneur appartienne à son pivot le plus proche.
Ensuite, cet algorithme est utilisé pour les conteneurs à deux feuilles, ce qui transforme la feuille en un nœud interne,et crée deux nœuds feuilles.
Le problème se pose lorsque les données arrivent après la fin de la création de l'index. Ce problème est dû à l'inadéquation inhérente de la partition de l'espace intercalaire avec le facteur de chevauchement entre les régions. Par conséquent, nous devons les insérer dans l'arbre tout en gardant la structure équilibrée avec un chevauchement minimum. Par conséquent, nous calculons le taux de chevauchement du nouvel ensemble avec tous les arbres existants dans le nœud de Fog, puis, nous choisissons la plus grande valeur du taux trouvé avec l'arbre correspondant, en fonction de la valeur trouvée nous aidons à insérer les objets dans l'ancien arbre ou de créer un nouvel index.

Algorithm 3 Construction of BCCF*-Tree

Input S : set of data, N node
Output N node

if $N = \perp$ **then**
  N= Creation_of_node(S)
else
  for $v \in S$ **do**
    if $d(N.p_1, v) \leq d(tree.p_2, v)$ **then**
      $N.sac_1.append(v)$
      if $sac_1 > c_{max}$ **then**
        $N.left$ = Creation_of_node(sac_1)
      end if
    else
      $sac_2.append(v)$;
      if $sac_2 > c_{max}$ **then**
        $N.right$ = Creation_of_node(sac_2)
      end if
    end if
  end for
end if
return N

4.5 Recherche kNN dans l'arbre BCCF*

Dans cette section, nous décrivons un algorithme de recherche permettant de répondre aux requêtes de type kNN (Chapitre). La procédure de recherche dans notre structure

(Section 4.4) est composée de deux étapes :

- **La première étape :** Estimation de l'ensemble d'arbres la plus proche que l'objet recherché 4)

Algorithm 4 Estimation de l'arbre le plus proche de l'objet recherché

Input $Set - Tree$ set of Tree , r_q research object
Output $Near - Tree$ set of Tree

1: min_{index}=0
2: **for** $i \in$ [0,$Set - Tree$] **do**
3: $distance$=d($Set - Tree[i].centre$,r_q)
4: **if** $distance \leq \min_{index}$ **then**
5: $min - index = i$
6: **end if**
7: **end for**$Near - Tree$.append($Set - Tree[i]$)
8: **for** $voisin \in Set - Tree[i]$.voisin **do**
$Near - Tree$.append($Set - Tree[voisin]$)
9: **end for**
10: **return** $Near - Tree$

- **La deuxième étape :** Lancer en parallèle l'algorithme 5 dans l'ensemble d'arbres estimé.

Algorithm 5 EstimationKNN

Input $Tree$:Tree , r_o : research object
Output r_q : query radius

1: r_q=∞
2: r_q = min(r_q, max((d($Tree.pivot1$, r_o) + $Tree$.r1), (d($Tree.pivot2$, r_o) + $Tree$.r2)))
3: rqGauche =∞
4: rqDroite = ∞
5: **if** $Tree.left$ IS NOT NULL **then**
6: EstimationKNN($Tree.left$,r_o, r_q)
7: **else**
8: rqGauche = min(r_q,(d($Tree.pivot1$, r_o) + $Tree.r1$))
9: **end if**
10: **if** $Tree.right$ IS NOT NULL **then**
11: EstimationKNN($Tree.right$,r_o, r_q)
12: **else**
13: rqDroite = min(r_q,(d($Tree.pivot2$, r_o) + $Tree.r2$))
14: **end if**
15: r_q = min(r_q, rqGauche, rqDroite)
16: **return** r_q

L'algorithme 5 commence avec un rayon de requête r_q initialisé à ∞ puis calcule la distance entre le point de requête q et les deux pivots $pivot_1$ et $pivot_2$ respectivement, tout en descendant dans l'arbre, en traversant l'index et en diminuant le rayon de requête.

Algorithm 6 RechercheKNN

Input $Tree$:Tree , r_o : research object, r_q : query radius k
Output $Near - vector$: set of vector

1: $Near - vector$=[]
2: **if** (d($Tree.pivot1$, r_o) + r_o) $\leq$ (Tree.r1+r_q) **then**
3: **if** $Tree.left$ IS NOT NULL **then**
RechercheKNN($Tree.left$, r_o, r_q,k)
4: **else**
$Near - vector$Gauche=Tri$_a nd_S election(Sac_1)$
5: **end if**
6: **end if**
7: **if** (d($Tree.pivot2$, r_o) + r_o) $\leq$ (Tree.r2+r_q) **then**
8: **if** $Tree.right$ IS NOT NULL **then**
RechercheKNN($Tree.right$, r_o, r_q,k)
9: **else**
$Near - vector$Gauche=Tri$_a nd_S election(Sac_2)$
10: **end if**
11: **end if**
12: $Near - vector$= Fusion($Near - vector$Gauche, $Near - vector$Droite)
13: **return** $Near - vector$

On trouve ensuite le rayon minimal de la requête en exécutant l'algorithme 6 qui parcourt l'arbre en incluant les nœuds feuilles, Les nœuds feuilles contiennent un sous-ensemble de données indexées dont le cardinal maximum est c_{max}. Pour trouver les plus proches voisins d'une feuille, il suffit de trier les données indexées en fonction de leurs distances croissantes à la requête q. Comme résultat de la recherche, les premiers objets triés dans une liste sont retournés.

L'algorithme global doit fusionner et trier les résultats de chaque arbre, puis sélectionner les k^e objets les plus proches.

4.6 Conclusion

Ce chapitre a été consacré à la partie conception, nous avons présenté la méthode de prétraitement (DBSCAN), la méthode de pré-indexation (ED) et la structure d'indexation proposée (BCCF*) pour gérer les données massives de l'IdO. L'objectif principal de cette architecture est de développer un nouveau mécanisme efficace pour le stockage et la récupération des données.

Chapitre 5

Implémentation et résultats

5.1 Introduction

Notre proposition vise à minimiser le coût de construction de BCCF et à répondre efficacement aux requêtes kNN, Dans ce chapitre, nous fournissons une évaluation expérimentale de cette structure. De plus, nous comparons cette structure avec les méthodes d'indexation existantes dans les espaces métriques, à savoir BCCF.

5.2 Collections indexées

Afin de démontrer l'efficacité de notre approche, nous avons réalisé toutes les expériences sur des données réelles avec des distributions de données sensiblement différentes afin de démontrer l'applicabilité étendue de notre indice. Deux jeux de données différents ont été utilisés pour les expériences et le tableau 5.1 ci-dessous présente quelques caractéristiques de ces jeux de données.

1. **Dataset 1 (Tracking)** : Le premier ensemble de données représente un ensemble de vecteurs de caractéristiques d'objets mobiles obtenus par un simulateur de suivi d'objets utilisant des caméras sans fil dans le multimédia réseau sans fil de capteurs lors d'une simulation aléatoire [83].
2. **Dataset 2 WARD (Wearable Action Recognition Database)** [85] : Le deuxième ensemble de données représente un base de données pour la reconnaissance d'activités humaines à l'aide de capteurs portables.

Dataset	**Nombre d'objets**	**Dimensions**	$C_{max} = \sqrt{n}$
Tracking	62702	20	250
WARD	1000000	5	316

Tab. 5.1 : Caractéristiques des datasets utilisées

5.3 Protocole des expérimentations

Dans cette partie nous avons lancé plusieurs expérimentations sur un prototype de l'arbre-BCCF* sous les conditions suivantes :

1. Nous utilisons consécutivement les deux collections introduites ci-dessus, de difficultés intrinsèques différentes.

5.4 Évaluation de la structure d'index

Dans cette section, nous examinons la qualité de l'index sur plusieurs critères, tels que le nombre de nœuds à chaque niveau et la distribution des objets, la hauteur de l'arbre, le nombre de nœuds internes, le nombre de conteneurs. Pour une bonne organisation, nous avons présenté les résultats de chaque base de données séparément.

5.4.1 Collection$_1$:Tracking

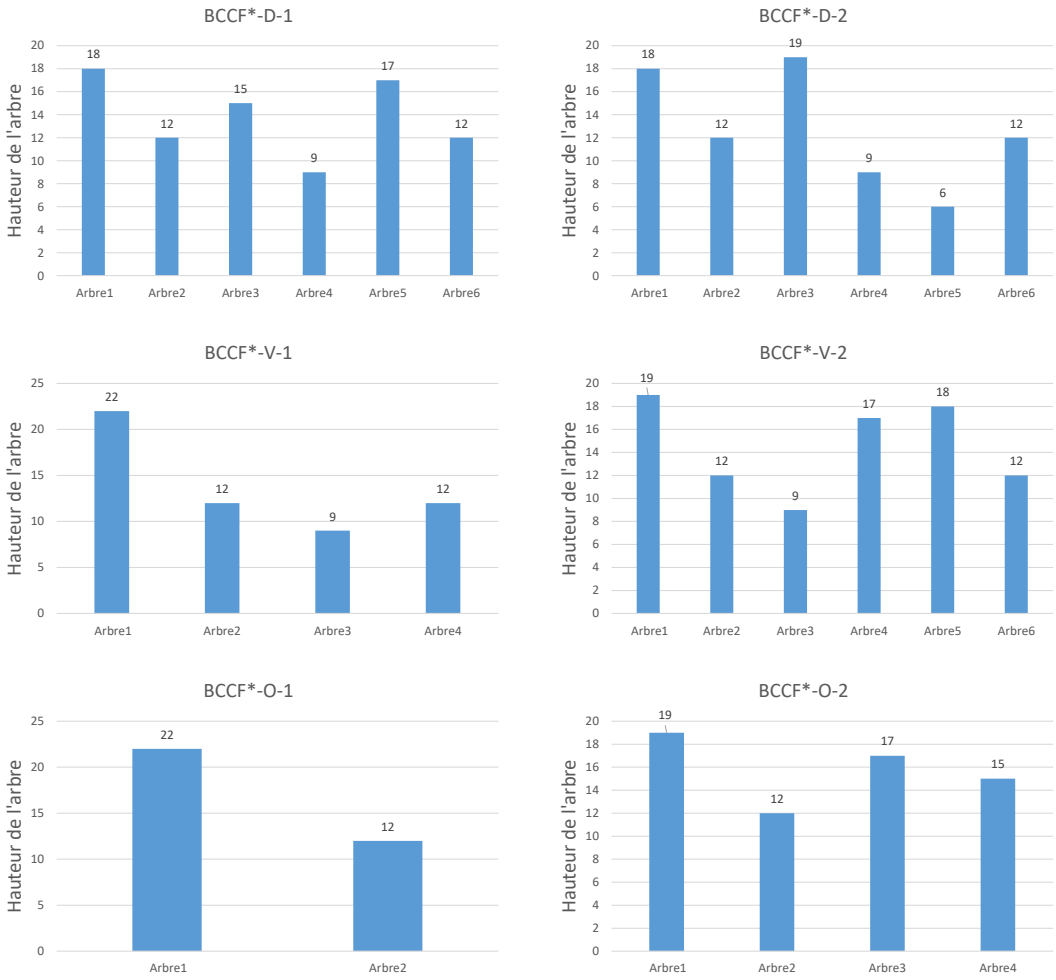


FIG. 5.1 : Hauteur des arbres "Ward"

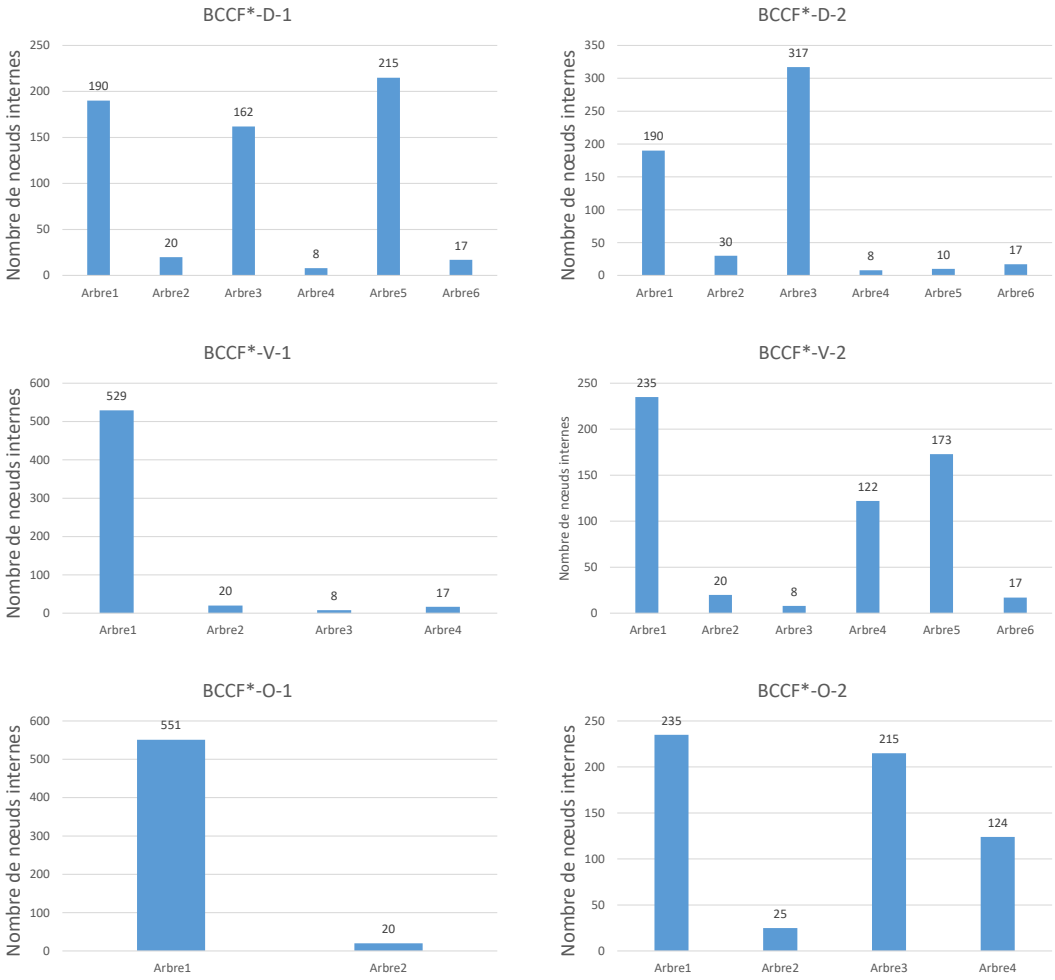


FIG. 5.2 : Nombre de nœuds internes des arbres "Ward"

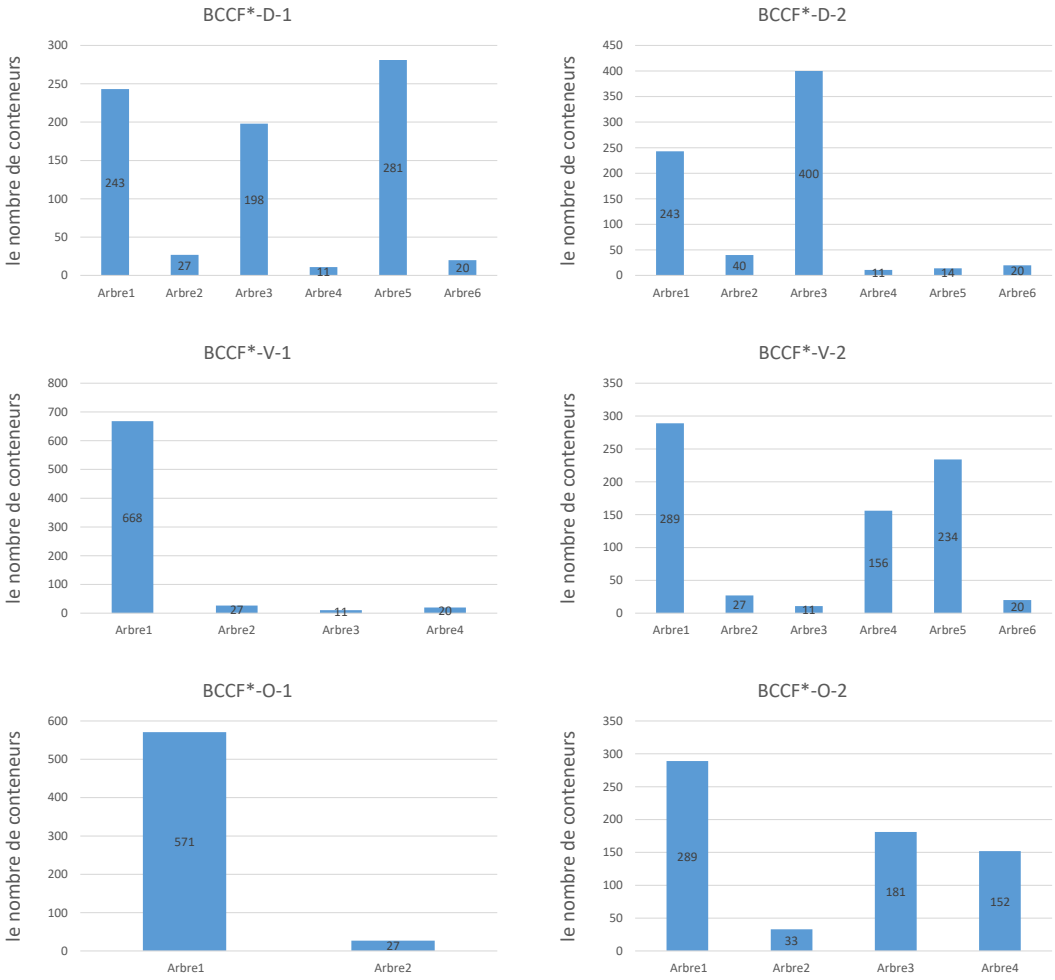


FIG. 5.3 : Nombre des conteneurs des arbres "Ward"

Les deux figures **??**, **??** résument les structures d'index de l'arbre-BCCF* sur ces deux collections.
Elles montrent en détail le nombre de nœuds créés à chaque niveau de l'arbre ainsi que la profondeur maximale atteinte par la structure.
Sans surprise, on constate que la plupart des structures BCCF* assurent un quasi-équilibre des index, le problème de dégénérescence ne se pose donc pas dans ce cas.

Nous nous intéressons ensuite à la répartition des objets dans l'index et au ratio de remplissage des nœuds feuilles.
Les figures **??**,5.1 montrent la hauteur de chaque arbre pour chaque version de BCCF* pour les deux collections.
Les figures **??** ,5.2 elles montrent les nombres des nœuds interne dans chaque arbre pour chaque version de BCCF* pour les deux collections.
Les figures **??**,5.3 montrent les nombres des conteneurs dans chaque arbre pour chaque version de BCCF* pour les deux collections.

Nous faisons les constatations suivantes :

- les valeurs des hauteurs sont proches donc il y a un équilibre de distribution

des objets dans la plupart des arbres.

- D'après les figures [**??**... **??**] et [**??** ... **??**] nous avons employé l'algorithme de construction basé sur un bon choix des pivots et un partitionnement de l'espace bien équilibré, nous ne pouvons raisonnablement pas nous attendre à ce que chaque feuille contienne exactement un nombre équitable des objets ; nous ne pouvons pas imaginer que l'équilibre sera parfait, même pas à ce qu'aucun nœud fils ne soit vide. Dans ce cadre, la distribution observée est alors très satisfaisante. Finalement, la profondeur observée dans les deux collections n'est pas plus de deux fois celle correspondant au cas parfait.

- Lors de la comparaison des hauteurs (les figures **??**, 5.1) avec des nœuds internes (les figures **??**, 5.2) et les nombres des conteneurs déduire un bon partitionnement des données.

5.5 Évaluation de l'algorithme de construction

Pour prouver l'efficacité de notre proposition BCCF*, nous la comparons avec la structure BCCF, en utilisant deux critères :

  – Le nombre de distances calculées.

  – Le nombre de comparaisons effectuées.

Nous avons expérimenté sur sept prototypes (les six versions BCCF* + BCCF) pour les ensembles de données précédents et avec les mêmes paramètres.

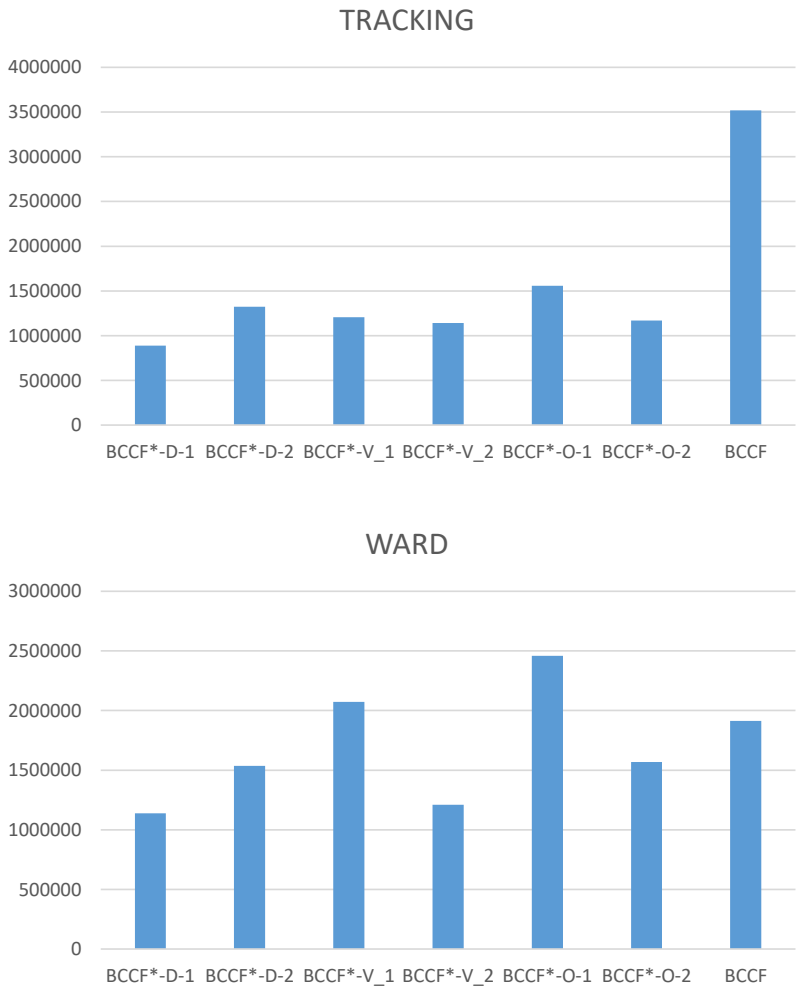


FIG. 5.4 : Nombre de distances calculées

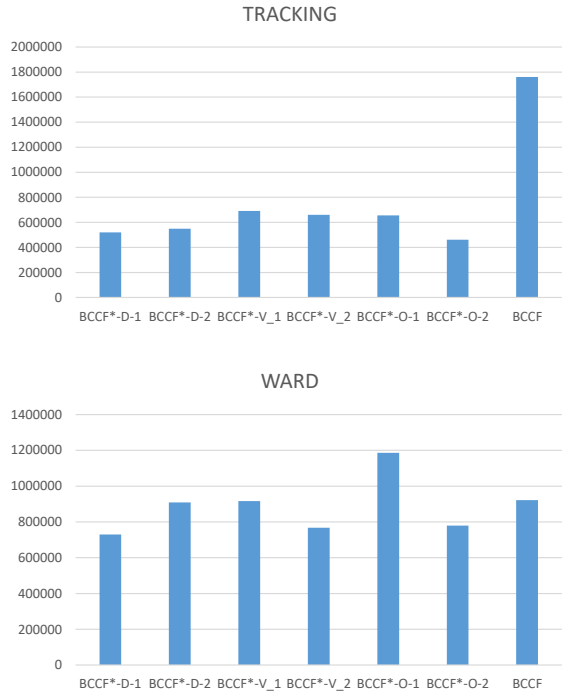


FIG. 5.5 : Nombre de comparaisons effectuées

Les figures 5.4,5.5 décrivent le nombre de calculs de distance et le nombre de comparaisons pour les différentes versions de l'index BCCF* présenter dans la table **??**.

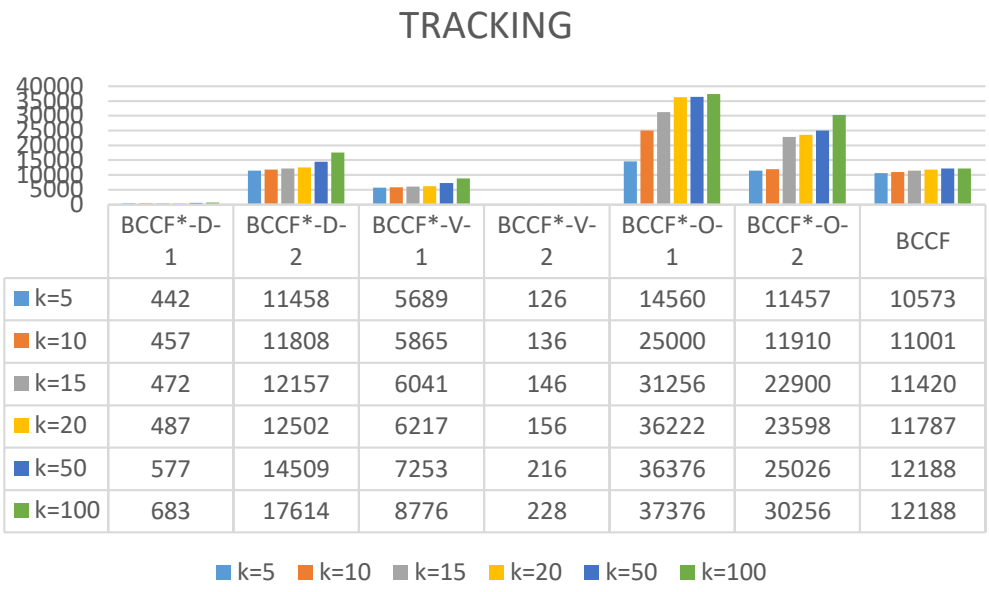
Dans les figures 5.4,5.5 nous constatons clairement que la proposition BCCF*-D-1 est la plus performante

5.6 Évaluation de la recherche kNN

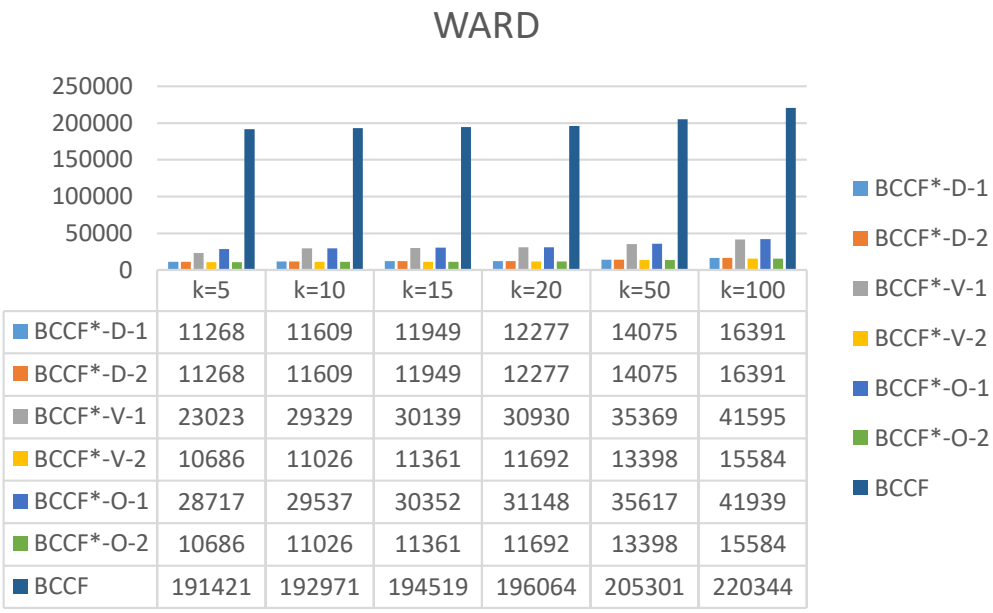
Nous sommes arrivés à la phase d'expérimentation de l'algorithme de recherche kNN. Afin de fournir une interprétation plus précise des résultats de nos expériences, nous rappelons que nous avons exécuté une centaine de requêtes kNN différentes sur les collections de données, et que nous avons fait la moyenne des résultats.

Nous mesurons :

- Le nombre moyen de distances calculées.
- Le nombre moyen de comparaisons effectuées.
- Le nombre moyen de nœuds feuilles visités lors de la recherche de kNN
- le temps exacte de la recherche.

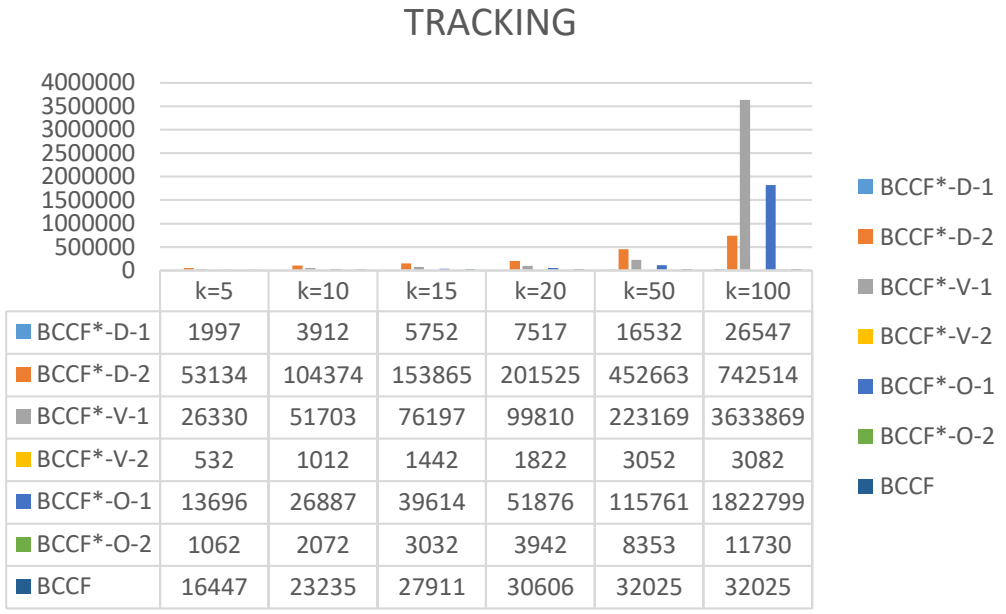


	BCCF*-D-1	BCCF*-D-2	BCCF*-V-1	BCCF*-V-2	BCCF*-O-1	BCCF*-O-2	BCCF
k=5	442	11458	5689	126	14560	11457	10573
k=10	457	11808	5865	136	25000	11910	11001
k=15	472	12157	6041	146	31256	22900	11420
k=20	487	12502	6217	156	36222	23598	11787
k=50	577	14509	7253	216	36376	25026	12188
k=100	683	17614	8776	228	37376	30256	12188

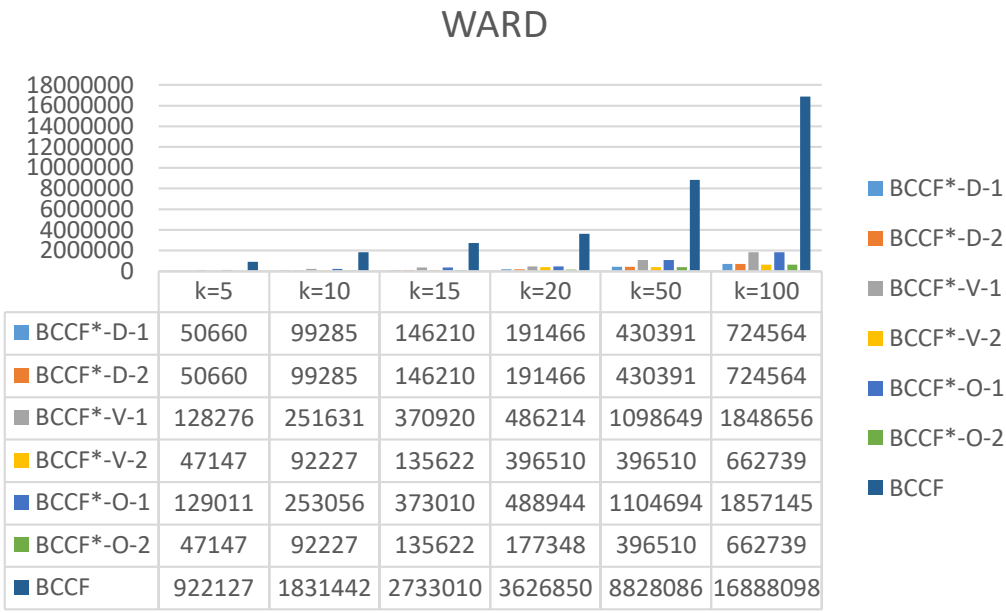


	k=5	k=10	k=15	k=20	k=50	k=100
BCCF*-D-1	11268	11609	11949	12277	14075	16391
BCCF*-D-2	11268	11609	11949	12277	14075	16391
BCCF*-V-1	23023	29329	30139	30930	35369	41595
BCCF*-V-2	10686	11026	11361	11692	13398	15584
BCCF*-O-1	28717	29537	30352	31148	35617	41939
BCCF*-O-2	10686	11026	11361	11692	13398	15584
BCCF	191421	192971	194519	196064	205301	220344

FIG. 5.6 : Nombre de distances calculées pour la recherche kNN



	k=5	k=10	k=15	k=20	k=50	k=100
BCCF*-D-1	1997	3912	5752	7517	16532	26547
BCCF*-D-2	53134	104374	153865	201525	452663	742514
BCCF*-V-1	26330	51703	76197	99810	223169	3633869
BCCF*-V-2	532	1012	1442	1822	3052	3082
BCCF*-O-1	13696	26887	39614	51876	115761	1822799
BCCF*-O-2	1062	2072	3032	3942	8353	11730
BCCF	16447	23235	27911	30606	32025	32025



	k=5	k=10	k=15	k=20	k=50	k=100
BCCF*-D-1	50660	99285	146210	191466	430391	724564
BCCF*-D-2	50660	99285	146210	191466	430391	724564
BCCF*-V-1	128276	251631	370920	486214	1098649	1848656
BCCF*-V-2	47147	92227	135622	396510	396510	662739
BCCF*-O-1	129011	253056	373010	488944	1104694	1857145
BCCF*-O-2	47147	92227	135622	177348	396510	662739
BCCF	922127	1831442	2733010	3626850	8828086	16888098

FIG. 5.7 : Nombre de comparaisons effectuées pour la recherche kNN

la figure 5.6 et la figure 5.7 montrent le nombre de distances et de comparaisons dans les cas de KNN $(k = 5, 10, 15, 20, 50, et 100)$ successivement.

En fonction de l'ensemble des résultats des figures précédentes, les conclusions suivantes peuvent être tirées :

- Sans exception, l'augmentation de la valeur de k augmente le coût de l'algorithme.
- Notons également, en comparant les méthodes entre elles, que l'arbre $\text{BCCF}^* - D - 1$ est le moins coûteux et l'arbre $\text{BCCF}^* - V - 1$ est le plus coûteux.

5.7 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle approche de l'indexation dans les espaces métriques, une méthode appelée "BCCF*-tree". Nous avons proposé des algorithmes de construction basés sur l'estimation du chevauchement et des méthodes de décision. Par la suite, nous avons proposé un algorithme de recherche des k plus proches voisins. Dans la première proposition de la phase de construction, l'arbre BCCF*-D-1 a montré ses performances par rapport aux autres versions de BCCF* et aussi par rapport à la version originale de BCCF.

Dans la deuxième partie, nous avons montré, également par des expériences, que l'algorithme de recherche des k plus proches voisins dans BCCF* dans les six versions est toujours le plus performant sur les deux collections utilisées dans cette étude.

Conclusion Générale

5.8 Conclusion

Cette étude a pour premier objectif d'optimiser la structure du BCCF en minimisant le coût de construction de l'index tout en garantissant la performance de l'indexation, ainsi que la flexibilité et la rapidité de la recherche d'information sur les données Ido.

Pour atteindre cet objectif, nous avons appliqué l'algorithme clustring à l'ensemble des données avant l'indexation En effet, la division de données à croissance exponentielle en sous-ensembles à l'aide de boules induit une dégénérescence de l'index en raison du décalage inhérent au partitionnement de l'espace.

La structure d'index que nous proposons a montré des résultats expérimentaux intéressants et compétitifs, tant dans la construction que dans la récupération de requêtes similaires utilisant le parallélisme lors de la traversée des nœuds d'index. Cela pourrait être une meilleure alternative pour l'indexation et la récupération des données Ido. Les perspectives que nous pouvons tirer de ce travail pour les travaux futurs sont les suivantes :

- Améliorer le temps des algorithmes de recherche tout en étant basé sur un environnement bien controlé, comme par exemple l'architecture peer to peer.
- Mener des études approfondies dans l'aspect mathématique des formes géométriques englobantes afin d'affiner ces dernières, et qui devront par la suite être simples pour programmer des algorithmes qui les gèrent.
- Mise en application de ces techniques dans des domaines précis.

Bibliographie

[1] Fatima Zahra Fagroud, Sanaa Elfilali, Hicham Toumi et al. "IOT et Cloud Computing : état de l'art". In : *Colloque sur les Objets et systèmes Connectés*. 2019.

[3] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In : *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012, p. 13-16.

[4] Weisong Shi et al. "Edge computing : Vision and challenges". In : *IEEE internet of things journal* 3.5 (2016), p. 637-646.

[5] Zineddine Kouahla. "Indexation dans les espaces métriques Index arborescent et parallélisation". Thèse de doct. Université de Nantes, 2013.

[6] Fabrizio Falchi, Claudio Gennaro et Pavel Zezula. "A content–addressable network for similarity search in metric spaces". In : *Databases, Information Systems, and Peer-to-Peer Computing*. Springer, 2006, p. 98-110.

[7] Luigi Atzori, Antonio Iera et Giacomo Morabito. "The internet of things : A survey". In : *Computer networks* 54.15 (2010), p. 2787-2805.

[8] Ovidiu Vermesan et al. "Internet of things strategic research roadmap". In : *Internet of things-global technological and societal trends* 1.2011 (2011), p. 9-52.

[9] Perry Xiao. *Designing Embedded Systems and the Internet of Things (IoT) with the ARM mbed*. John Wiley & Sons, 2018.

[10] Shruti G Hegde Soumyalatha. "Study of IoT : understanding IoT architecture, applications, issues and challenges". In : *1st International Conference on Innovations in Computing & Net-working (ICICN16), CSE, RRCE. International Journal of Advanced Networking & Applications*. 478. 2016.

[11] Rafiullah Khan et al. "Future internet : the internet of things architecture, possible applications and key challenges". In : *2012 10th international conference on frontiers of information technology*. IEEE. 2012, p. 257-260.

[13] Peter Mell, Tim Grance et al. "The NIST definition of cloud computing". In : (2011).

[14] K Chandrasekaran. *Essentials of cloud computing*. CrC Press, 2014.

[15] Mohammad Ubaidullah Bokhari, Qahtan Makki Shallal et Yahya Kord Tamandani. "Cloud computing service models : A comparative study". In : *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE. 2016, p. 890-895.

[17] Bhaskar Prasad Rimal et Mohamed A El-Refaey. "A framework of scientific workflow management systems for multi-tenant cloud orchestration environment". In : *2010 19th IEEE International Workshops on Enabling Technologies : Infrastructures for Collaborative Enterprises*. IEEE. 2010, p. 88-93.

[18] Hibatullah Alzahrani. "A brief survey of cloud computing". In : *Global Journal of Computer Science and Technology* (2016).

[19] Mereena Thomas. "A Review paper on BIG Data". In : *International Research Journal of Engineering and Technology (IRJET)* 2.09 (2015), p. 1030-1034.

[20] Achla Gupta, Soma Bandyopadhyay et SS Thakur. "Cloud computing : its characteristics, security issues and challenges". In : *Review of Computere Engineering Studies* 4.2 (2017), p. 76-81.

[21] Anca Apostu et al. "Study on advantages and disadvantages of Cloud Computing–the advantages of Telemetry Applications in the Cloud". In : *Recent advances in applied computer science and digital services* 2103 (2013).

[22] Matthew Finnegan. "Boeing 787s to create half a terabyte of data per flight, says Virgin Atlantic". In : *Computerworld UK* 6 (2013), p. 1-2.

[24] Shanhe Yi et al. "Fog computing : Platform and applications". In : *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*. IEEE. 2015, p. 73-78.

[25] Subhadeep Sarkar, Subarna Chatterjee et Sudip Misra. "Assessment of the Suitability of Fog Computing in the Context of Internet of Things". In : *IEEE Transactions on Cloud Computing* 6.1 (2015), p. 46-59.

[26] Mithun Mukherjee et al. "Security and privacy in fog computing : Challenges". In : *IEEE Access* 5 (2017), p. 19293-19304.

[27] Mung Chiang et Tao Zhang. "Fog and IoT : An overview of research opportunities". In : *IEEE Internet of things journal* 3.6 (2016), p. 854-864.

[28] Yuan Ai, Mugen Peng et Kecheng Zhang. "Edge computing technologies for Internet of Things : a primer". In : *Digital Communications and Networks* 4.2 (2018), p. 77-86.

[29] Yang Liu, Jonathan E Fieldsend et Geyong Min. "A framework of fog computing : Architecture, challenges, and optimization". In : *IEEE Access* 5 (2017), p. 25445-25454.

[30] Flavio Bonomi et al. “Fog computing : A platform for internet of things and analytics”. In : *Big data and internet of things : A roadmap for smart environments*. Springer, 2014, p. 169-186.

[31] Goiuri Peralta et al. “Fog computing based efficient IoT scheme for the Industry 4.0”. In : *2017 IEEE international workshop of electronics, control, measurement, signals and their application to mechatronics (ECMSM)*. IEEE. 2017, p. 1-6.

[33] Mohamed Firdhous, Osman Ghazali et Suhaidi Hassan. “Fog computing : Will it be the future of cloud computing ?” In : The Third International Conference on Informatics & Applications (ICIA2014). 2014.

[35] Mahadev Satyanarayanan. “How we created edge computing”. In : *Nature Electronics* 2.1 (2019), p. 42-42.

[36] Wazir Zada Khan et al. “Edge computing : A survey”. In : *Future Generation Computer Systems* 97 (2019), p. 219-235.

[37] Bin Han et al. “Context-awareness enhances 5G multi-access edge computing reliability”. In : *IEEE Access* 7 (2019), p. 21290-21299.

[38] Sachchidanand Singh. “Optimize cloud computations using edge computing”. In : *2017 International Conference on Big Data, IoT and Data Science (BID)*. IEEE. 2017, p. 49-53.

[39] Volker Gaede et Oliver Günther. “Multidimensional access methods”. In : *ACM Computing Surveys (CSUR)* 30.2 (1998), p. 170-231.

[40] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In : *Communications of the ACM* 18.9 (1975), p. 509-517.

[41] Mohamed Amine Ferrag et al. “Big IoT Data Indexing : Architecture, Techniques and Open Research Challenges”. In : *2019 International Conference on Networking and Advanced Systems (ICNAS)*. IEEE. 2019, p. 1-6.

[42] Caetano Traina et al. “Slim-trees : High performance metric trees minimizing overlap between nodes”. In : *International Conference on Extending Database Technology*. Springer. 2000, p. 51-65.

[43] Jörg P Bachmann. “The SuperM-Tree : Indexing metric spaces with sized objects”. In : *arXiv preprint arXiv :1901.11453* (2019).

[44] Shichao Jin, Okhee Kim et Wenya Feng. “M X-tree : A Double Hierarchical Metric Index with Overlap Reduction”. In : *International Conference on Computational Science and Its Applications*. Springer. 2013, p. 574-589.

[45] Stefan Berchtold, Daniel A Keim et Hans-Peter Kriegel. “The X-tree : An index structure for high-dimensional data”. In : *Very Large Data-Bases*. 1996, p. 28-39.

[46] Lu CHEN et al. “Pivot-based Metric Indexing.(2017)”. In : *Proceedings of the VLDB Endowment : 43rd International conference, Munich Germany, 2017 August 28-September*. T. 1.

[47] Gonzalo Navarro et Nora Reyes. “New dynamic metric indices for secondary memory”. In : *Information Systems* 59 (2016), p. 48-78.

[48] Safia Brinis, Caetano Traina et Agma JM Traina. “Hollow-tree : a metric access method for data with missing values”. In : *Journal of Intelligent Information Systems* 53.3 (2019), p. 481-508.

[49] Peter N Yianilos. “Data structures and algorithms for nearest neighbor search in general metric spaces”. In : *Soda*. T. 93. 194. 1993, p. 311-21.

[50] Tolga Bozkaya et Meral Ozsoyoglu. “Distance-based indexing for high-dimensional metric spaces”. In : *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. 1997, p. 357-368.

[51] Ives Rene Venturini Pola, Caetano Traina et Agma Juci Machado Traina. “The mm-tree : A memory-based metric tree without overlap between nodes”. In : *East European Conference on Advances in Databases and Information Systems*. Springer. 2007, p. 157-171.

[52] Zineddine Kouahla et al. “XM-tree : data driven computational model by using metric extended nodes with non-overlapping in high-dimensional metric spaces”. In : *Computational and Mathematical Organization Theory* 25.2 (2019), p. 196-223.

[53] Weiguo Wan et Hyo Jong Lee. “Deep feature representation and balltree for face sketch recognition”. In : *International Journal of System Assurance Engineering and Management* (2019), p. 1-6.

[54] Mohamad Dolatshah, Ali Hadian et Behrouz Minaei-Bidgoli. “Ball*-tree : Efficient spatial indexing for constrained nearest-neighbor search in metric spaces”. In : *arXiv preprint arXiv :1511.00628* (2015).

[55] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[56] Jeffrey K Uhlmann. “Satisfying general proximity/similarity queries with metric trees”. In : *Information processing letters* 40.4 (1991), p. 175-179.

[57] Christos Faloutsos et King-Ip Lin. “FastMap : A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets”. In : *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*. 1995, p. 163-174.

[58] James McNames. “A nearest trajectory strategy for time series prediction”. In : *Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling*. KU Leuven Belgium. 1998, p. 112-128.

[59] Christian Merkwirth, Ulrich Parlitz et Werner Lauterborn. “Fast nearest-neighbor searching for nonlinear signal processing”. In : *Physical Review E* 62.2 (2000), p. 2089.

[60] Sergey Brin. “Near neighbor search in large metric spaces”. In : (1995).

[61] Zineddine Kouahla et Adeel Anjum. “A Parallel Implementation of GHB Tree”. In : *IFIP International Conference on Computational Intelligence and Its Applications*. Springer. 2018, p. 47-55.

[62] Lu Chen et al. “Efficient metric indexing for similarity search and similarity joins”. In : *IEEE Transactions on Knowledge and Data Engineering* 29.3 (2015), p. 556-571.

[63] Zineddine Kouahla et José Martinez. “A new intersection tree for content-based image retrieval”. In : *2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI)*. IEEE. 2012, p. 1-6.

[64] Ives Renê Venturini Pola, Caetano Traina Jr et Agma Juci Machado Traina. “The NOBH-tree : Improving in-memory metric access methods by using metric hyperplanes with non-overlapping nodes”. In : *Data & Knowledge Engineering* 94 (2014), p. 65-88.

[65] Keyu Yang et al. “Distributed similarity queries in metric spaces”. In : *Data Science and Engineering* 4.2 (2019), p. 93-108.

[66] Shi-guang Liu et Yin-wei Wei. “Fast nearest neighbor searching based on improved VP-tree”. In : *Pattern Recognition Letters* 60 (2015), p. 8-15.

[67] Il’ya Markov. “VP-tree : Content-based image indexing”. In : *Proceedings of the Spring Young Researcher’s*. T. 7. 2007, 00268a.

[69] DT-Tri Nguyen et al. “An index scheme for similarity search on cloud computing using mapreduce over docker container”. In : *Proceedings of the 10th International Conference on ubiquitous information management and communication*. 2016, p. 1-6.

[70] Tri DT Nguyen et Eui-Nam Huh. “An efficient similar image search framework for large-scale data on cloud”. In : *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*. 2017, p. 1-8.

[71] Hua Cheng et al. “Distributed indexes design to accelerate similarity based images retrieval in airport video monitoring systems”. In : *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE. 2015, p. 1908-1912.

[72] Claudia Deco et al. “XM-Tree, a new index for Web Information Retrieval”. In : *Journal of Computer Science & Technology* 8 (2008).

[73] Ting Liu, Andrew W Moore et Alexander Gray. “Efficient exact k-NN and nonparametric classification in high dimensions”. In : *Proceedings of the 16th International Conference on Neural Information Processing Systems*. 2003, p. 265-272.

[74] Bastian Leibe, Krystian Mikolajczyk et Bernt Schiele. “Efficient clustering and matching for object class recognition.” In : *BMVC*. 2006, p. 789-798.

[75] Kouahla Zineddine, Ferrag Mohamed Amine et Anjum Adeel. “Indexing Multimedia Data with an Extension of Binary Tree–Image Search by Content–”. In : *International Journal of Informatics and Applied Mathematics* 1.1 (), p. 47-55.

[76] Phuc Do et Trung Hong Phan. “A Distributed M-Tree for Similarity Search in Large Multimedia Database on Spark”. In : *Handbook of Research on Multimedia Cyber Security*. IGI Global, 2020, p. 146-164.

[77] Suman Saha. “Community Detection in Complex Network Metric Space Nearest Neighbor Search Low Rank Approximation and Optimality”. In : (2020).

[78] CHAI KAH HIENG. *HARDWARE BASED ACCELERATOR FOR DATABASE QUERY USING M-TREE*. 2018.

[79] Dilek Küçük Matci et Uğur Avdan. “Comparative analysis of unsupervised classification methods for mapping burned forest areas”. In : *Arabian Journal of Geosciences* 13.15 (2020), p. 1-13.

[80] Henrique Batista da Silva et al. “Video similarity search by using compact representations”. In : *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 2016, p. 80-83.

[81] Tomás Skopal, Jaroslav Pokornỳ et Vaclav Snasel. “PM-tree : Pivoting Metric Tree for Similarity Search in Multimedia Databases.” In : *ADBIS (Local Proceedings)*. 2004, p. 803-815.

[82] Gang Chen et al. “Metric similarity joins using MapReduce”. In : *IEEE Transactions on Knowledge and Data Engineering* 29.3 (2016), p. 656-669.

[83] Ala-Eddine Benrazek et al. “An efficient indexing for Internet of Things massive data based on cloud-fog computing”. In : *Transactions on emerging telecommunications technologies* 31.3 (2020), e3868.

[84] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In : *kdd*. T. 96. 34. 1996, p. 226-231.

Webographie

[2] *Internet des objets*. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. (date d'accès : 08/07/2021).

[12] Luis M Vaquero et al. *A break in the clouds : towards a cloud definition*. 2008.

[16] *Les types de services Cloud*.

[23] *EDGE*. https://www.rtinsights.com/10-ways-fog-computing-extends-the-edge/. Accessed : 2021-08-11.

[32] *Fog Computing and the Internet of Things : Extend the Cloud to Where the Things Are. White Paper. 2016.Available online :* https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. Accessed : 2021-08-11.

[34] *What is edge computing ?, howpublished = https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/, note = Accessed : 2021-07-15*.

[68] Marcus Adamsson et Aleksandar Vorkapic. *A comparison study of Kd-tree, Vp-tree and Octree for storing neuronal morphology data with respect to performance*. 2016.

[85] *bdd*. https://people.eecs.berkeley.edu/yang/software/WAR/WARD1.zip,2019. Accessed : 2021-07-11.