الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

A distributed indexing mechanism for the Internet of Things

Encadré Par :

Dr. KOUAHLA Zineddine

Présenté par :

Abdi Charaf-Eddine

Juillet 2019

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Dr. KOUAHLA Zineddine for the continuous support of my master final study project and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor.

Besides my advisor, I would like to thank the Ph.D student BENRAZEK Alaeddine, for his help, encouragement, and continuous support; I appreciate all his efforts with me considering him my second advisor, teacher, friend, and a brother.

My sincere thanks also goes to the LABSTIC laboratory for the internship and all the researchers working there who helped me even with discussions, ideas, comments, critics...

Finally, I would like to thank all my teachers and apreciate the big favor they did for me to get here.

Abstract

Over the years, the growing innovations in information and communication technologies (ICT) in the computer and communications industry have allowed distributed systems researchers to design new systems and models to meet the needs of an increasingly diverse user community. In this project, we studied a set of indexing techniques in spatial metrics with a strict subset. We had neither the time nor the space to make an exhaustive overview. As a result, we were able to highlight only the metric space indices. Based on what appear to be the best recent approaches, we proposed a distributed approach to indexing data in a metric space, essentially inspired by the binary tree. In summary, our proposal can be considered as a paginated and parameterized version of the latter. Nevertheless, the problem encountered each time by these techniques is that of the "curse of the dimension". In a second step, we developed a distributed version to improve performance, which, in our opinion, should stagnate, or at least progress in this family of indices.

Contents

List of Figures

List of Tables

General Introduction

Over the years, the growing innovations in **I**nformation and **C**ommunication **T**echnologies (ICT) in the computer and communications industry have enabled distributed systems researchers to design new systems and models to meet the needs of an increasingly diverse user community. In particular, the emergence of the ***I**nternet **o**f **T**hings (IoT)* allows a large number of virtual and physical objects and devices with identification, detection and network capabilities to communicate and interact seamlessly with each other and with other resources (e.g., devices, services) in the network (Internet) [3].

Today, *IoT* is more of a descriptive term for a vision where everything must be connected to the Internet. IoT will be essential in the future because the new concept opens up opportunities for new services and innovations. *Cisco* in 2011 [4] studied the increase in the number of objects connected to the Internet and found that by 2020, there will be 50 billion objects connected, or 7 objects for each person. As a result of this increase, the amount of digital data that is continuously captured by these objects has exploded. This expansion has created the need to store, manage and secure huge volumes of data to make the most of this mass of information for an increasingly large audience. To meet these challenges, a new IT paradigm called *Cloud Computing* is needed, in which data and services are available in a flexible and transparent way.

Cloud Computing is a new paradigm, which offers a variety of Internet services, high storage capacity, high processing capacities and greater data security. With these services, the *Cloud* is able to cover the weakness of traditional systems that have limited storage and processing capacity. Despite all these advantages, *Cloud Computing* presents a high latency problem when transmitting raw, complex and massive sensor data to the cloud

due to the long distance between them, which negatively affects real-time applications such as video surveillance systems, patient monitoring systems and vehicle control systems. In this case, effective *indexing* and *research* in large data collections is, therefore, one of the most pressing current problems.

This project examines existing indexing techniques in the literature and external systems that rely on load balancing mechanisms with trigger post-ingestion to propose a new distributed method *indexing* based on the paradigm *Cloud Computing*. The main objective is to exploit their advantages, which are represented by the large processing and storage capacity, to index the massive data IoT in order to distribute the input data on the distributed structure during indexing for minimal re-balancing.

Memory Organization

The reminder of the project includes the state of the art of *Internet of Things*, the description of indexing techniques in the *Internet of Things*, our conceptual proposal of the indexing method, as well as its implementation and finally, the results obtained.

- **Chapter 1** concerns the *Internet of Things and Cloud Computing*. In this chapter we define the *Internet of Things*, we present their architecture and infrastructure, then we discuss its advantages and disadvantages and describe the different application areas. Concerning *Cloud Computing*, we present its definition and its types, delivery models. Then, we discuss its advantages and disadvantages.
- **Chapter 2** focuses on the Indexing Techniques. In this state of the art, we present different metric space *indexing* techniques and their algorithms for the construction and search for nearest neighbours that exist in the literature.
- **Chapter 3** presents our contribution, which is the design of a new distributed *indexing* technique that adapts the *Cloud Computing* paradigm to index the *Internet of Things* data to improve the quality of indexes and the efficiency of construction and search algorithms.

- The proposal was validated on different types of real data, with experimental results presented in **Chapter 4**. We evaluated our method in two parts, the first part is a centralized version and the second part is a distributed version.

This project concludes with a general conclusion that summarizes the essential contribution of our work and presents some research perspectives.

Chapter 1

Internet of Things and Cloud Computing

1.1 Introduction

The revolution in Information and Communication Technology has created the concept of the *Internet of Things (IoT)* which is considered as the extension of the Internet network [5], widespread in different domains. This emerging network allows billions of objects which can be physical or virtual (car, Glasses, Watches, Smart-Phone, etc...) to be connected and communicated between them through the Internet in an intelligent way using standard communication protocols. This large number of objects produces continuously a huge amount of data that requires significant computational power and incredible storage capacity for analysis and storage. To satisfy these requirements, the adaptation of *Cloud Computing* could be profitable [6], which would facilitate the use of their highest processing performance and storage capacity [7].

In this chapter we present a state of the art on the *Internet of Things* as well as the *Cloud Computing* paradigm in two part, where in the first part, we will review the concepts and definitions of the state of the art related to the *Internet of Things*, its architecture and infrastructure, the advantages and disadvantages and its applications domains. Thereafter, in the second part, we apply the same for *Cloud Computing*.

1.2 Internet of Things (IoT)

1.2.1 Definition

Even nowadays, it is still difficult to find a clear and accurate definition of the *Internet of Things (IoT)* so we gathered some recent definitions extracted from the literature.

Definition 1 *: "Internet of Things (IoT) is an integrated part of* ***future*** *Internet and could be defined as a dynamic global network infrastructure with self configuring capabilities based on standard and practical communication protocols where physical and virtual 'things' have identities, physical attributes and virtual personalities. Plus, they use intelligent interfaces and are seamlessly integrated into the information network. In the IoT, 'things' are expected to become active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information 'sensed' about the environment, while reacting autonomously to the 'real/physical world' events and influencing it by running processes that trigger actions and create services with or without direct human intervention. Interfaces in the form of services facilitate interactions with these 'smart things' over the Internet, query and change their state and any information associated with them, taking into account security and privacy issues" [8].*

Definition 2 *: "The Global Standards Initiative on Internet of Things (IoT-GSI) in Recommendation ITU-T Y.2060 defined the Internet of Things (IoT) as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [9].*

Definition 3 *: "A network of networks that allows, via normalized and unified electronic identification systems and mobile wireless devices, to directly and unambiguously identify digital entities and physical objects and therefore to recover, store, transfer and process the related data, without discontinuity between the physical and virtual worlds, [10].*

Definition 4 *: "Things having identities and virtual personalities in operation in smart areas exploiting intelligent interfaces to connect and communicate among social, environmental, and user contexts". A different definition, that puts the main target on the seamless integration, can be formulated as "Interconnected objects having an active role in what could be known as the future Internet". The semantic origin of the expression is composed by two words and concepts: "Internet" and "Thing", where "Internet" can be defined as "The world-wide network of interconnected computer networks, based on a standard communication protocol, the Internet suite (TCP/IP)", whereas "Thing" is "an object not exactly identifiable" thus, semantically, "Internet of Things" means that "a world-wide network of interconnected objects unambiguously addressable, based on standard communication protocols" [11].*

Definition 5 *: "The basic idea of this concept is that the pervasive presence around us of a variety of things or objects – like Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with one another and collaborate with their neighbors to achieve common goals" [12].*

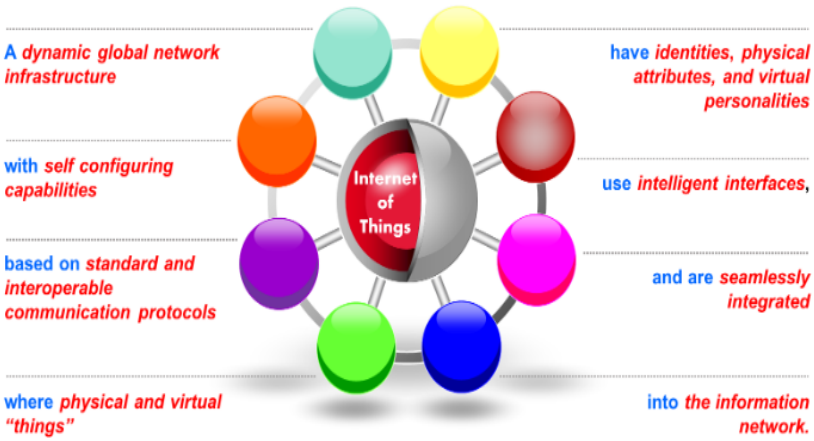


Figure 1.1: IERC iot Definition [1].

From our point of view and according to our understanding, we see that *"the Internet of Things is a network of objects generally wireless sensors and actuators, which are uniquely identifiable in the world unambiguously, operating in an intelligent environment. These objects can interact and communicate with each other and with the environment through*

standard communication protocols using information and communication technology for the exchange of data and information that is stored in the Cloud and processed by Big-Data techniques".

1.2.2 Architecture

Today's Internet uses TCP/IP protocol stack for communication between network hosts that was suggested in the past. However, the IoT connects billions of objects which will create much larger traffic and much more data storage is needed [13]. Also, IoT can face several challenges specially associated with privacy and security [14].

Thus, the new proposed architecture for IoT has to address several factors like scalability, interoperability, reliability, QoS, etc. Since IoT connects everything and everyone to exchange information among themselves, the trafc and storage in the network will also increase in the exponential way. Thus, IoT development depends on the technology progress and style of varied new applications and business models. The basic architecture of the IoT is proposed in [15], this architecture is known by the 3-layer architecture where it consists of the perception layer, the network layer and the application layer, as shown on the left of Figure 1.2. The authors in [13] propose the reference architecture of the loT, which is known as the 5-layer architecture that consists of the Business layer, the Application layer, the Processing layer, the Transport layer and the Perception layer, as shown on the right of Figure 1.2, thanks to the technological architecture of the Internet, the logical structure of the Telecommunications Management Network and the combination of the network characteristics of the objects. These layers are briefly described below:

1. *Perception Layer*: The Perception layer is additionally referred to as 'Device Layer'. It consists of the physical objects and sensing element devices. The sensors are often RFID, 2D-barcode, or Infrared sensor depending upon objects identication method. This layer basically deals with the identication and collection of objects specic information by the sensor devices. Depending on the sort of sensors, the data can be about temperature, humidity, vibration, orientation, acceleration, motion, location, chemical changes within the air etc. The collected data is then passed to

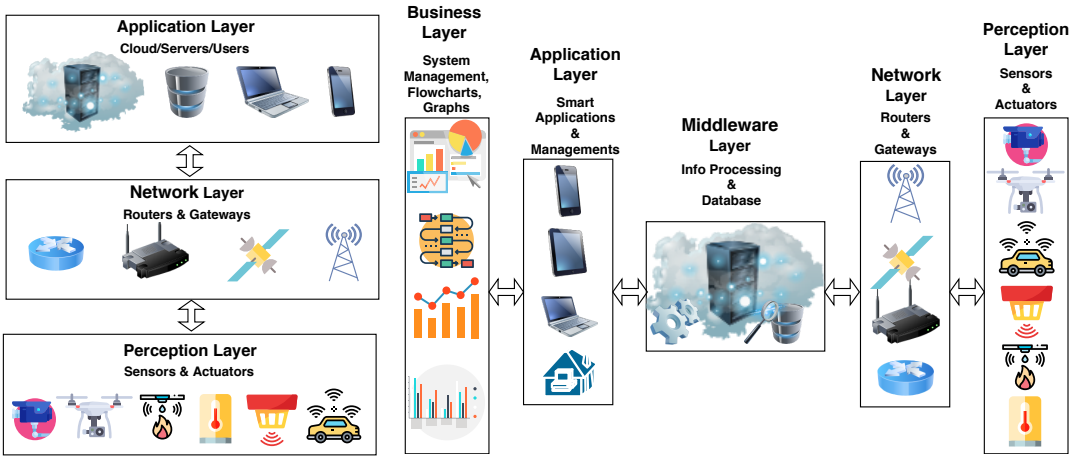


Figure 1.2: 3 and 5 Layer Architectures of the IoT

Network layer for its secure transmission to the data processing system.

2. *Network Layer*: The Network layer also can be known as "*Transmission Layer*". This layer securely transfers the data from sensor devices to the data processing system. The transmission medium may be wired or wireless and technology can be Wi, UMTS, Infrared, Bluetooth, ZigBee, RFID, 4G/5G/LTE, Satellite, etc depending upon the sensor devices. Thus, the Network layer transfers the data from Perception layer to Middleware layer.

3. *Middleware Layer*: The devices of the IoT implement totally different kind of services. Each device connects and communicates with solely those alternative devices that implement a similar service kind. This layer is chargeable for the service management and has link to the database. It receives the data from Network layer and store within the database. It performs data processing and omnipresent computation and takes automatic decision based on the results.

4. *Application Layer*: This layer provides global management of the application based on the objects information processed in the Middleware layer. The applications implemented by IoT can be Smart Home, Smart City, Smart Health, Smart Farming, Intelligent Transportation, Smart Industry, etc.

5. *Business Layer*: This layer is chargeable for the management of overall IoT system as well as the applications and services. It builds business models, owcharts, graphs,

etc., based on the information received from Application Layer. The real success of the IoT technology additionally depends on the great business models. Based on the analysis of results, this layer can be facilitated to work out the long run actions and business methods.

1.2.3 Infrastructure

In our framework the Infrastructure is the minimum of elements that are necessary to realize an IoT-based application.

- ***The objects*** included in IoT are not only complex devices such as mobile phones, but they also include everyday objects [16], these physical objects have embedded sensor, intelligence and connectivity technologies, allowing them to communicate with other objects using module component manufacturers, who provide connectivity to the objects via embedded hardware and software components. These players include electronics engineers, semiconductor foundries, sensor manufacturers and embedded software developers providing connectivity.[17]

- ***Network*** types that immediately come to mind are Bluetooth, Wifi and 3G/4G. Wifi and Bluetooth have a limited range. This does not make them useless, quite the opposite. In a restricted environment with a good supply of energy sources. 3G/4G is more far-reaching, but again requires a lot of energy, but is easy to implement.[18]

- ***Cloud*** operators mainly providing storage, analysis and processing of raw data. In this segment, the traditional Internet and server players are competing with new players deploying their own IT infrastructures [17].

- ***Interfaces*** software suppliers, or middleware, to communicate the different objects. This segment includes traditional software publishers. [17].

- ***Security*** actors, present at all levels of the chain, from object design to services. At best, these IT security actors work closely together with all actors in the value chain. Some are also absorbed by the Internet of Things actors. [17].

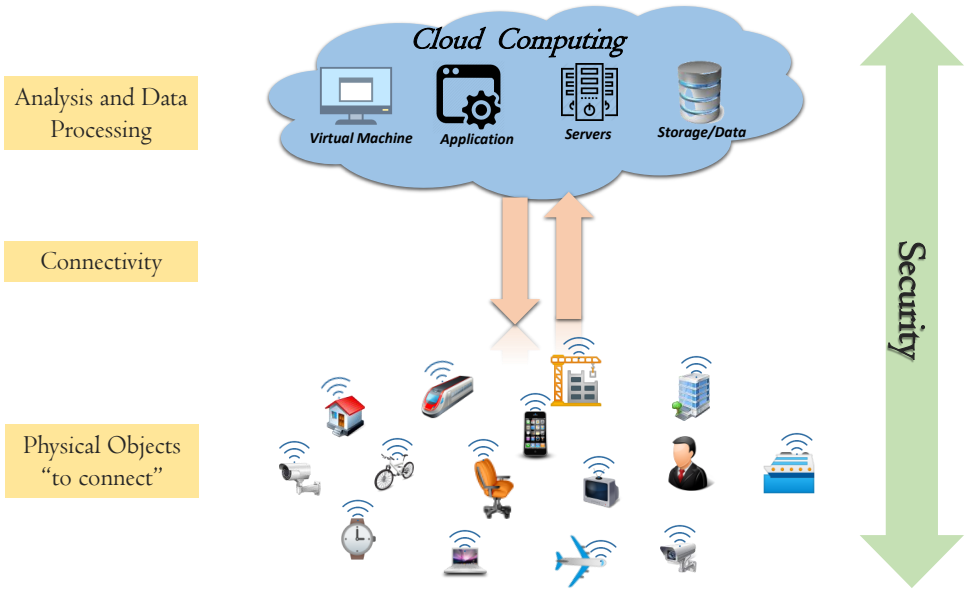


Figure 1.3: IoT Infrastructure.

1.2.4 Advantages and Disadvantages

As any new technology, it always has two sides, a good and a bad one: [19]

1. **Advantages:**

   Better Quality of Life: IoT applications will increase comfort and much better management in our daily life. Therefore, they aim to improve life quality.

   Automation and Control: Without human involvement, machines are automating and controlling large amount of data, that leads to faster outputs.

   Communication: Since IoT is based on communication between devices, in which physical devices are able to remain connected and therefore full transparency is available with less inefficiency and better quality.

   Monitoring Saves Money and Time: Since IoT uses intelligent sensors to monitor various aspects of our daily lives for various applications, saving time and money.

Better Environment: Conserves natural resources and trees and helps to make the planet greener and better.

New Business Opportunities: Creates new business for IoT technology, which will increase economic processes and create new jobs.

2. **Disadvantages:**

Complexity: The IoT is a diverse and complex network. Any failure or bugs within the software system or hardware can have serious consequences. Even power outage will cause important inconvenience.

Privacy/Security: IoT involves multiple devices and technologies and, therefore, multiple companies focus on security. Since a lot of information associated with the context are going to be transmitted by the smart sensors, there's a high risk of losing private information.

Compatibility: As devices from different manufacturers will be interconnected in IoT, presently, there is no international standard of compatibility for the tagging and monitoring equipment.

Technology Controls Life: Our lives are going to be more and more controlled by technology, and can be addicted to it. The younger generation is already hooked in to technology for every small thing. With IoT, this dependency can unfold among generations and in daily routines of users. We have to make a decision about how much of our daily lives are we willing to mechanize and be controlled by technology.

Lesser Employment of Menial Staff: With the appearance of technology, daily activities are becoming automated by utilizing IoT with less human intervention, that successively causes fewer needs of human resources. This causes unemployment is-

sue in the society.

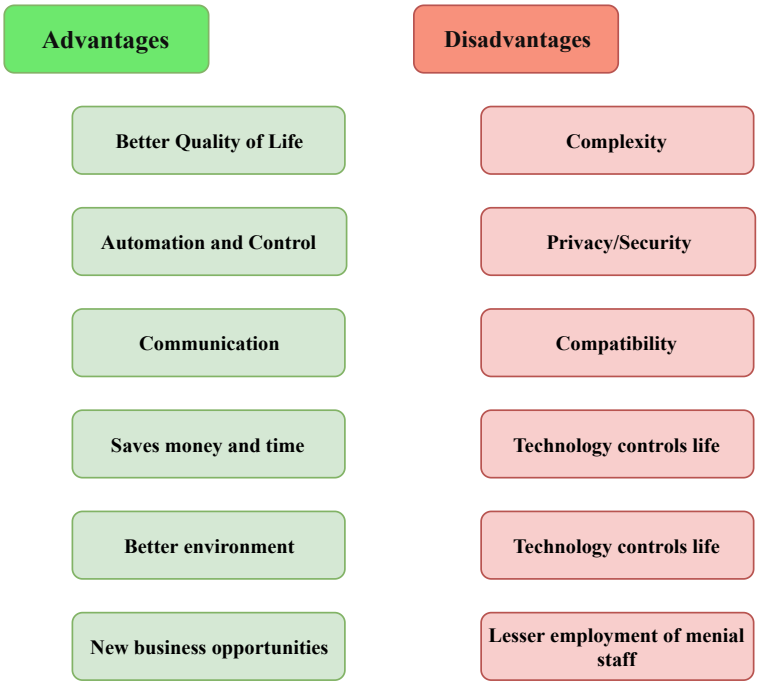


Figure 1.4: Advantages and Disadvantages.

1.2.5 Application Domains

IoT will find its applications in almost all aspects of our lives. Below are some examples [20].

- *Prediction of Natural Disasters:* The mixture of sensors and their autonomous coordination and simulation can facilitate the prediction of the prevalence of landslides or various natural disasters and the taking of appropriate measures in advance.
- *Industry Applications:* The IoT will be nd applications in the industry, for example, managing a eet car for a company. IoT helps to monitor their environmental performance and process data to determine and select those that require maintenance.
- *Water Scarceness Monitoring:* IoT will facilitate the discovery of water scarcity in completely different places. Sensor networks, linked to relevant simulation activi-

ties, can not only monitor future water-related interventions, such as drainage area management, but can even be used to alert stream users, for example, if an upstream event, such as an accidental discharge of wastewater into the stream, could have dangerous impacts.

- *Design of Smart Homes:* IoT can help in the design of smart homes, for example, energy management, interaction with household appliances, emergency detection, home safety and nding things easily, home safety, etc.

Figure 1.5: IoT Application Domains.

- *Medical Applications:* IoT can also be used in the medical sector to save lives or improve the quality of higher education. health parameter monitoring, surveillance activities, independent living support, medication monitoring, etc.

- *Agriculture Application:* A network of matching sensors will detect the information, process it and inform the farmer using a communication infrastructure, e. g. a text message by mobile phone on the portion of the field that requires special attention. It can be an intelligent packaging of seeds, fertilizers and mildew

management mechanisms that meet indigenous conditions and indicate the measures to be taken. An intelligent agricultural system will help agronomists to better understand plant growth patterns and agricultural practices by understanding soil conditions and climate variability. This will increase agricultural productivity by avoiding inappropriate operating conditions.

- *Intelligent Transport System Design:* Intelligent transport can offer efcient control and management transport with the use of advanced sensor, information and network technology. Intelligent transport will have several fascinating options such as electronic non-stop road tolling, mobile emergency command and control, transport law enforcement, monitoring vehicle rule violations, reducing environmental pollution, anti-theft systems, traffic congestion prevention on trafc, incident reporting on trafc, intelligent beaconing, reducing arrival delays etc.

- *Design of Smart Cities:* IoT can help to design smart cities, for example by monitoring air quality, discovering emergency routes, broadcasting lighting from the city, watering gardens, etc.

- *Smart Metering and Monitoring:* The IoT style for intelligent metering and monitoring can facilitate the encouragement of correct automatic meter reading and the sending of invoices to customers. IoT can also be used to design such a scheme for the maintenance and remote monitoring of wind turbines, gas, water and environmental metering and monitoring.

- *Smart Security:* The IoT can also be used for security and surveillance applications in eld, such as space monitoring, people and property tracking, infrastructure and equipment maintenance, alarms, etc.

1.3 Cloud Computing

1.3.1 Definition

Cloud Computing idea is not a new one; John McCarthy has already predicted that computing facilities are going to be provided to the public use as a utility giving it the

similarity between telephony and computing. *"..computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."* John McCarthy, speaking at the MIT Centennial in 1961 [21].

Certainly, there still a lack of a standard definition of Cloud computing. For this reason, recently there has been work on standardizing the definition of Cloud computing. As an example, the work in [22] compared over 20 different definitions from a variety of sources to confirm a standard definition. in this work we'll adopt the National Institute of Standards and Technology (NIST) definition of Cloud computing as it covers all aspects of it.

"The NIST Definition of Cloud Computing: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model is composed of five essential characteristics, three service models, and four deployment models" [23].

1.3.2 Cloud Computing Types

There are different types of Cloud computing, each with its own advantages and drawbacks [24]:

Public Cloud : A Cloud in which service suppliers provide their resources as services to the general public. Public Clouds offer several key benefits to service providers, including no initial capital investment on infrastructure and shifting of risks to infrastructure providers. However, public Clouds lack fine-grained control over data, network and security settings, which hampers their effectiveness in many business scenarios.

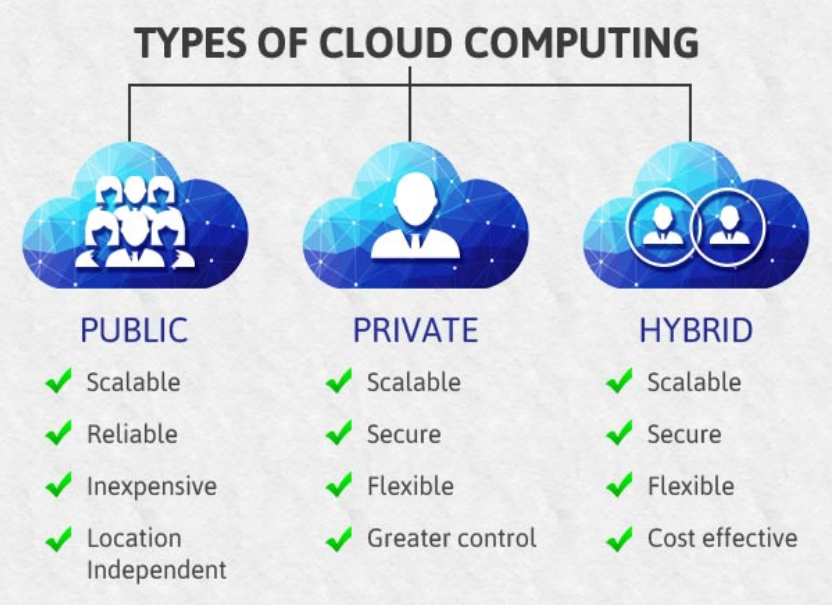


Figure 1.6: Types of Cloud Computing [2]

Private Cloud : Also referred to as internal Clouds, private Clouds are designed for exclusive use by one organization. A private Cloud could also be built and managed by the organization or by external providers. A private Cloud offers the best degree of control over performance, reliability and security. However, they're typically criticized for being almost like ancient proprietary server farms and don't give advantages like no up-front capital costs.

Hybrid Cloud : A hybrid Cloud is a combination of public and private Cloud models that tries to deal with the restrictions of every approach. In a hybrid Cloud, part of of the service infrastructure runs privately Clouds while the remaining part runs publicly Clouds. Hybrid Clouds provide additional flexibility than each public and private Clouds. Specifically, they provide tighter control and security over application data compared to public Clouds, while still facilitating on-demand service expansion and contraction. On the down side, designing a hybrid Cloud needs carefully determining the most effective split between public and private Cloud elements.

1.3.3 Cloud Computing Delivery Models

The Cloud computing can be categorized in three types according to the types of delivery models as showed in Fig.1 1.7 [25], namely Infrastructure as a service (IaaS), Software as a service (SaaS) and Platform as a service (PaaS).

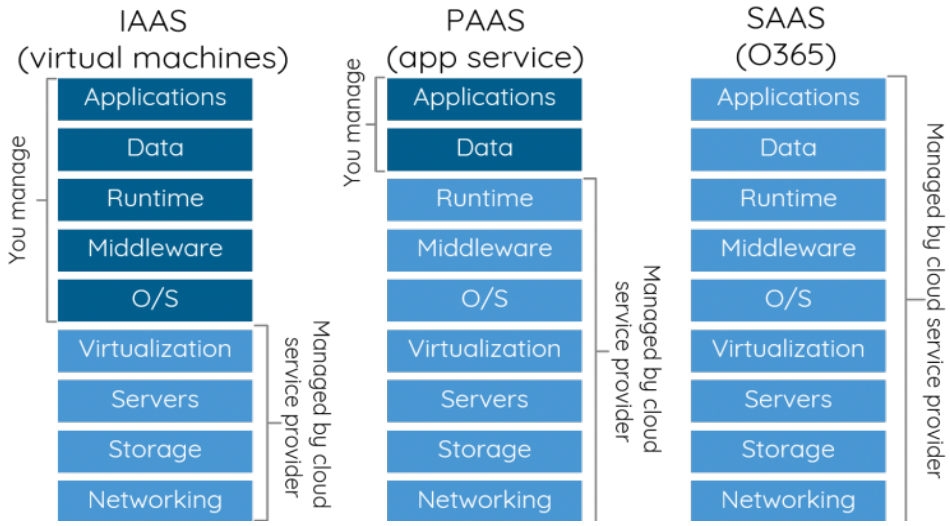


Figure 1.7: Cloud Computing Delivery Models.

Infrastructure as a Service (IaaS) :

IPs manage an oversized set of computing resources, like storing and processing capacity. Through virtualization, they're able to split, assign and dynamically re-size these resources to build ad-hoc systems as demanded by customers, the SPs. They deploy the software system stacks that run their services. This is the Infrastructure as a Service (IaaS) situation.

Platform as a Service (PaaS) :

Cloud systems offers a further abstraction level:instead of supplying a virtualized infrastructure, they can provide the software platform where systems run on. The size of the hardware resources demanded by the execution of the services is created in a clear manner. This is denoted as Platform as a Service (PaaS). A well-known example is the Google Apps Engine.

Software as a Service (SaaS) :

Finally, there are services of potential interest to a large kind of users hosted in Cloud systems. This is an alter-native to locally run applications. An example of this can be the online alternatives of typical office applications like word processors. This scenario is named software as a Service (SaaS).

1.3.4 Advantages and Disadvantages

1. **Advantages:**

   Reduce Management Costs: Managing IT infrastructures or deploying enterprise software applications for employees' desktops creates an administration surcharge, which is an important factor in the total cost to the company's business. The Cloud reduces this surcharge by relieving the problems of installing, managing and maintaining hardware, server operating systems, and deploying the application.

   Access Anywhere: Cloud services are web-based and do not depend on the used computer . Documents and applications hosted in the Cloud can be accessed from anywhere.

   Virtually Unlimited IT Resources: Resources such as computing power and data storage space are available on demand as needed, providing a high level of flexibility and scalability to meet the company's changing needs. We too often forget the notion of saturation of machines and processors.

   Energy-Efficient: Some arguments for making the Cloud an energy-efficient IT solution are as follows: Customers share a set of IT resources; Suppliers use larger, more modern and more energy-efficient data centers Increased server utilization due to server virtualization.

   Facilitating Group Collaboration: For many users, this is an important advantage

of the Cloud when several users can easily collaborate on documents and projects.

2. **Disadvantages:**

   Bandwidth can increase the budget: The bandwidth that would be required to put this in the Cloud is huge and the costs would be so high that it is more advantageous to buy the storage ourselves than to pay someone else to do it.

   Internet Dependency: In the absence of a connection, we no longer have access to services.

   Technical Issues: While it is true that data and information on the cloud can be accessed at any time and from anywhere, there are times when the system will experience serious malfunctions. Companies must be aware that this technology is generally exposed to breakdowns and various technical problems. Even the most efficient cloud service providers face these types of problems, despite maintaining high maintenance standards [26].

   Inflexibility: Choosing a Cloud computing vendor often means locking the business into using their proprietary applications or formats. For instance, it's not possible to insert a document created in another application into a Google Docs spreadsheet. Furthermore, a corporation must be able to add and/or take off Cloud computing users as necessary as its business grows or contracts [26].

   Security: The other major issue of Cloud is delineated by security. Before adopting this technology, beneficiaries ought to recognize that they're going to be surrendering all their company's sensitive data to a third-party Cloud service supplier. This could doubtless impose a great risk to the company. Hence, business need to make sure that they choose the most reliable service provider, who will keep their information totally secure [26].

1.4 Conclusion

In this chapter we presented a state of the art of the Internet of Things, as well as of the Cloud Computing paradigm. In the following chapter we will describe the concept of data indexation and present the main structures proposed in recent years.

Chapter 2

Indexing Techniques

2.1 Introduction

In this chapter, We will present a state of the art of the indexing techniques. Several authors have proposed reference works or syntheses on the subject[27, 28, 29]. Since this work focuses on general metric spaces and not multidimensional spaces, we limited ourselves to the contributions and lessons that can be drawn from the study of the main proposals in the literature, in the case of tree indexes.

2.2 Background

For several decades, the amount of data has increased considerably, like in multimedia (images, audio and video), but also time series, fingerprints, DNA sequences, among others. This means that the use of effective and scalable tools is essential. Efficient indexing is an increasingly important area in computer science. Indexing techniques have been improved to deal with searches on large collections of data. However, it has been found that the indexing processes become more difficult. It is difficult to compare these techniques [30, 31, 32], their effectiveness depend on different factors (type of data, quality of the computing machine, etc.). The researchers in this field have proposed several algorithms for fast information retrieval. The reader can find several surveys to present and compare existing multidimensional indexing techniques [27, 28, 29]. At the same time, the objects to be indexed are often more complex than mere vectors (e.g., sets, graphs) or cannot be

simply and meaningfully concatenated to give a larger vector (e.g., color histograms and keywords)[33, 34]. Hence, the focus of indexing has partly moved from multidimensional spaces to metric spaces. In this work, we are interested in finding an effective technical indexing similarity to reduce the number of distance calculations required to answer queries. It is important to emphasize that the usefulness of information not only depends on its quality, but also on how fast it is recovered, which, in return, depends on how it is indexed. Formally, a metric space is defined for a family of elements that are comparable through a given distance. The distance function measures the dissimilarity between two elements from a given database, in such a way that smaller distances correspond to more similar elements. Let $\mathcal{O}$ be a set of elements. Let $d : \mathcal{O} \times \mathcal{O} \rightarrow R^+$ be a distance function, which verifies:

- Non-negativity: $\forall (x, y) \in \mathcal{O}^2, d(x, y) \geq 0$,
- Reflexivity: $\forall x \in \mathcal{O}, d(x, x) = 0$,
- Symmetry: $\forall (x, y) \in \mathcal{O}^2, d(x, y) = d(y, x)$,
- Triangle inequality: $\forall (x, y, z) \in \mathcal{O}^3, d(x, y) + d(y, z) \leq d(x, z)$.

The concept of metric space is rather simple and leads to a limited number of possibilities for querying an actual database of such elements. These are called similarity queries and several variants exist. There are two main types of similarity queries: the range and the k-nearest neighbor queries.

Metric spaces introduce the notion of topological ball, which is close to a broad match. It allows distinguishing between inside and external objects.

Based on these two partitioning techniques, we can introduce a short taxonomy, as shown in Figure 2.1, which represents several indexing techniques in metric spaces. There are two main cases:

2.2.1 Non-partitionning space class

The first class does not enforce a partitioning of the space. We find essentially the M-tree family. The M-tree [35] builds a balanced index, allows incremental updates. Unfortunately it suffers from the problem of overlapping that increases the number of distance calculations to answer a query. Slim-tree [36] is an optimized version of M-tree. It mainly reorganizes the M-tree in order to reduce overlaps. The used algorithm has shown good performances on the research algorithm and has reduced its processing time, its defect is the need to reinserting the objects. In the context of the reorganization of objects in compact clusters, Almeida [37] proposed a new structure but just for an approximate search, called Divisive-Agglomerative Hierarchical Clustering or DAHC-tree. It is constructed by dividing and grouping the data set into compact clusters. In [38], the authors proposed an extension of Slim-Tree named Slim*-tree, that exploits the best properties from ball and the BST as a hash function to search within a bucket file. The aim is to minimize the overlap between the shapes. Thus, this technique proposes the reinsertion in the same level of the tree of a saturated page (node) before to split it. This reinsertion avoids splitting. It therefore ensures a continuous reorganization of the tree. The problem has not been resolved and the reinsertion of objects remains costly on a large scale. Although the response time has been improved compared to the Slim-tree. A novel clustering based dynamic indexing and retrieval approach is proposed, termed as CD-Tree [32], updates the structure with constant insertion of data. The nodes in the CD-Tree are fitted by Gaussian Mixture Models. In our opinion, the problem is not totally solved because the update of construction phase remains slow and becomes costly on a large scale.

It is important to mention that there are other techniques which are not based on the hierarchical structure, like SAT, which is based on Voronoi diagrams [39, 40, 41].

2.2.2 Partitionning space class

The second class is based on the partitioning of the space. There have been a number of longitudinal studies, such as [42, 43, 44].

Two sub-approaches are included: the first uses ball partitioning, like VP-tree [45, 46, 47], mVP-tree [48]. In this methods, the choice of the pivots plays a very important role

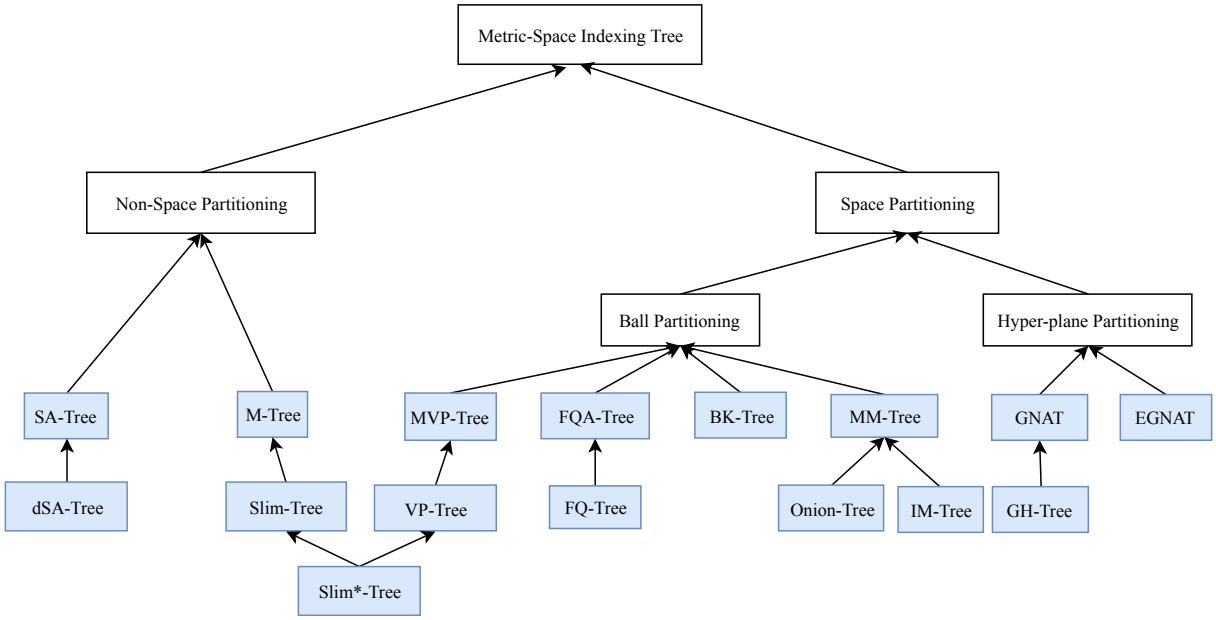


Figure 2.1: A simplified taxonomy of indexing techniques

on the index structure. As a result, Yianilos proposed the VP-tree [45] , which is based on finding the median element of a set of objects. The mVP-tree is a generalization of the VP-tree, the nodes are divided into quantiles.

The second approach uses hyperplan partitioning, like GH-tree. It has proven its efficiency in some dimension$*s, but it is still inefficient in large dimensions. The principle of this technique$

This principle of partitioning eliminates the problem of overlapping between shapes. However, in this type of approach, a problem arises in cases where a demand point is close to the border between two regions; it is necessary to visit all the neighboring regions which makes the index less efficient.

2.3 Recent works

Recently, a few techniques have emerged in this second class; combine two trees to improve the search time, an idea that has been proposed by Ryan Curtin[49], it uses the kd-tree and ball-tree to take advantage of both information. Several difficulties were cited by the authors. The main problem is that the efficiency decreases if the dimension is greater than 10. Other techniques [41] have been proposed in the last five years trying to index large-scale data but does not meet the exact but approximate queries, and other try to

compress the index [42, 50]. This leaves the door open to other proposals in the future. Combining two index, Voronoi-shaped with ball-shaped regions, was the basis for NOBH-trees[51], that allows a non-overlapping division of the data space. The problem with this technique lies in the complexity of the enclosing forms. This increases the cost of insertion and search operations.

2.4 Conclusion

In this chapter, We presented a state of the art of the indexing techniques. Several methods were discussed. On the basis of two indexing techniques, partitioning and non-partitioning, we can introduce a short taxonomy of some indexing techniques in metric spaces. First, we noted, as several authors have already pointed out, that metric spaces have become a popular model for circumventing the limitation of vector spaces in different applications. We can also analyze the main factors that affect the efficiency of search algorithms in metric spaces

Chapter 3

Analysis and Conception

3.1 Introduction

In this chapter, we present our proposed indexing structure for Internet of Things data management at the Cloud Computing level. Our structure is based on the following concepts:

- ***Tree Index:*** First of all, we chose the indexing of the trees structure. This may be a bias, but the approach to indexing in multidimensional spaces in the literature is largely dominant and almost exclusive in metric spaces. In the absence of a satisfactory solution, to our knowledge, and in this vast field of as yet imperfect knowledge, technologies and techniques, we wish to address, if not the resolution, then at least the improvement of the indexing of metric data spaces.

- ***The partitioning of the metric space using spheres:*** There are several advantages to conducting searches in a metric space. The most important thing is that more types of data from the Internet of Things can be indexed, because the approach is based only on calculating distances between objects and not on their content. The sphere is a topological notion that specifies the ball in space. In our work, the space is recursively divided into two spheres (two sub-parts, "left" and "right") as shown in Fig. 3.1.

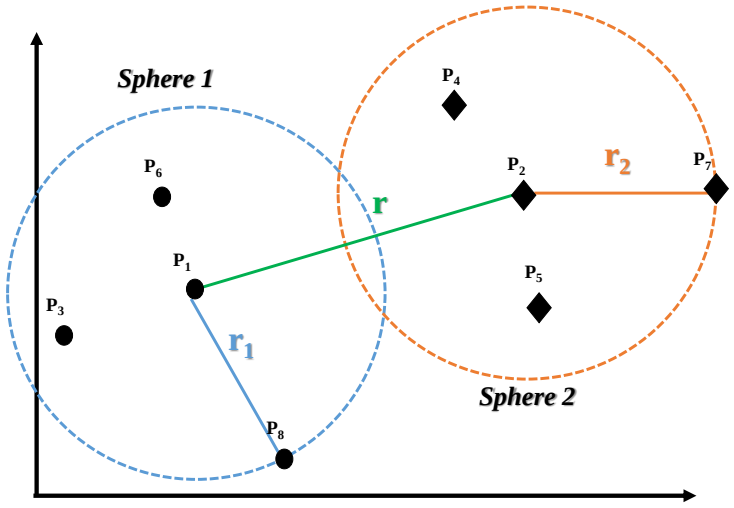


Figure 3.1: Example of Space Partitioning with Spheres.

- ***Distribution Index:*** the distribution of the index is necessary to create an efficient, balanced, flexible and extensible indexing system that supports an unlimited number of clients. In our work, the index structure is hierarchically distributed as shown in Figure 3.2.

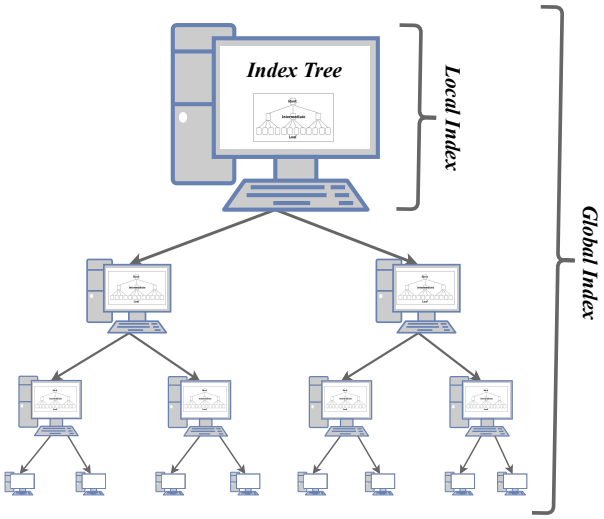


Figure 3.2: Hierarchical Distribution of the Index.

3.2 System Overview

The main objective of the proposed method is to develop a new effective indexing mechanism for the IoT environment. This mechanism adapts and leverages the Cloud Com-

puting paradigm to improve the quality of research and user queries on large scale IoT data. Figure 3.3 shows a global view of the IoT-Cloud Computing architecture for the proposed IoT data indexing method and also show the main elements that are included in the proposed system with the roles of each in indexing and discovering IoT data process which are discussed in [3].

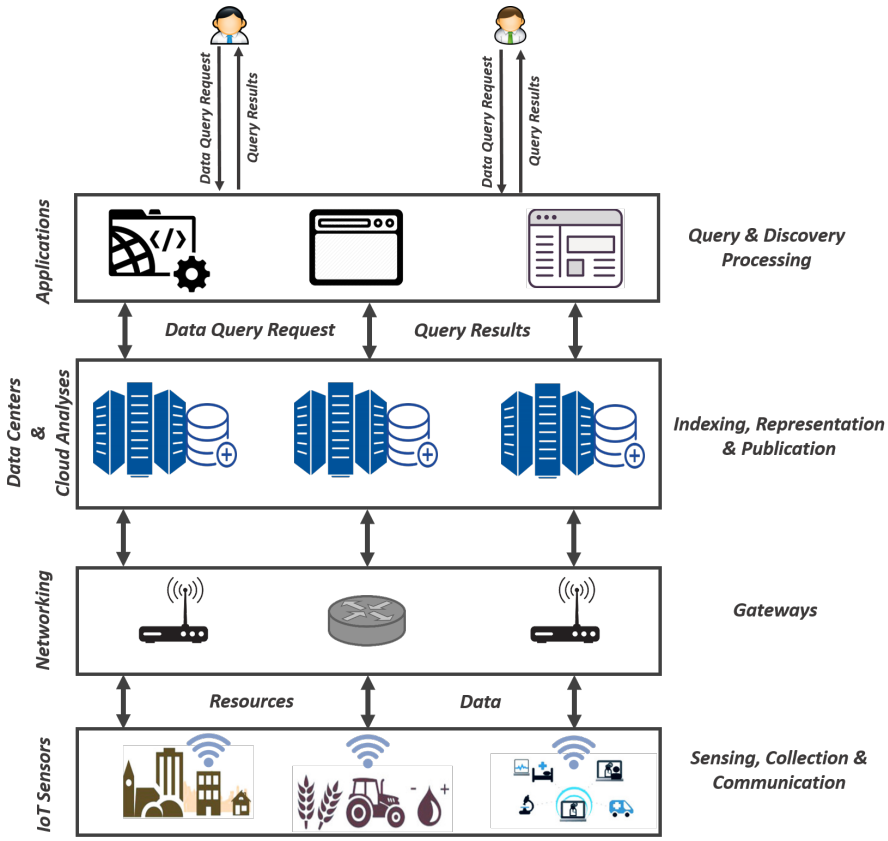


Figure 3.3: System Overview.

The scenario of the proposed system is started in the **Connected Objects layer** or in **the IoT Sensors layer**. The role of this layer in the IoT data indexing and discovery process is : (a) *detection*, (b) *collection*, and (c) *communication*. This layer consists of millions of heterogeneous sensors connected of different domains (industrial and commercial, health, smart cities, transport, farmers, environment, etc.), these sensors will produce a large amount of information on the monitored environment. This information transmitted to the upper layer in Cloud Computing.

The role of the **Cloud computing layer** is : (a) *representation*, (b) *publication* and

(c) *indexing*, in addition to : (d) *Storage* and (e) *Analysis*. The observation data received by the lower layer (IoT Sensors layer) is raw digital data. The Cloud will represent this data in formats that can be understood by the machine. Then, will index this data in the proposed indexing structure.

For **Application layer**, their role is : (a) *Discovery*, (b) *Query*. The real data is indexed and stored in the Cloud thanks to its large storage capacity and can be used for long-term analysis. In reality, the Cloud Computing layer is not the source layer of the request, actually are the users who use the applications inside the Application layer, but the request transmits from the Application layer by the user to the Cloud Computing layer, and the latter is responsible for processing this request. After receiving the request in the Cloud layer of the Application layer, the Cloud computing start the search process (exact or similarity search, depending on the nature of the application) on the proposed indexing structure, then, sends the answer to the user (the Application layer).

3.3 Construction

3.3.1 Without Index

To see the difference, and before we get to our advanced methods, we should pass by the most basic one, and that is done by storing all collected data randomly as it came from the sensors and actuators network. This method consists storing data in a container one after one, first in first stored without any condition or any partitioning algorithm. the most important reason for this method is to breakdown our coming advanced methods in subsections 3.3.3 and 3.3.4 .

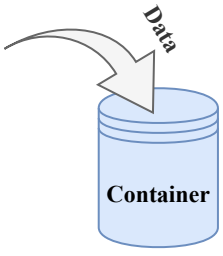


Figure 3.4: Storing Data Without Index

3.3.2 Binary Tree (B-tree)

This method is more structured by using a Binary Tree with dual pivots, the data inserted are stored in the pivots of the tree. The left and right pointers points to its corresponded sub-tree, that have been selected based on the euclidean distance between the inserted vector and the pivots of the current node (see Fig 3.5).

Algorithm 1 represents the construction recursive function of the binary tree. That creates the node if its null, and insert the two first vectors in the pivots if they are empty, else it travel through the tree until finding the appropriate leaf node to insert it according to the euclidean distance function between the inserted object and the pivots of the internal nodes.

Algorithm 1 : BinaryInsertion (Tree, V_x)

```
1: if Tree is not Null then
2:    if Tree.P1 is Null then
3:       Tree.P1=V_x
4:    else if Tree.P2 is Null then
5:       Tree.P2=V_x
6:    else
7:       if Distance(Tree.P1, V_x) < Distance(Tree.P2, V_x) then
8:          Tree.Left = BinaryInsertion(Tree.Left, V_x)
9:       else
10:         Tree.Right = BinaryInsertion(Tree.Right, V_x)
11:      end if
12:   end if
13: else
14:   Tree = Noeud()
15:   Tree = BinaryInsertion(Tree, V_x)
16: end if
17: return Tree
```

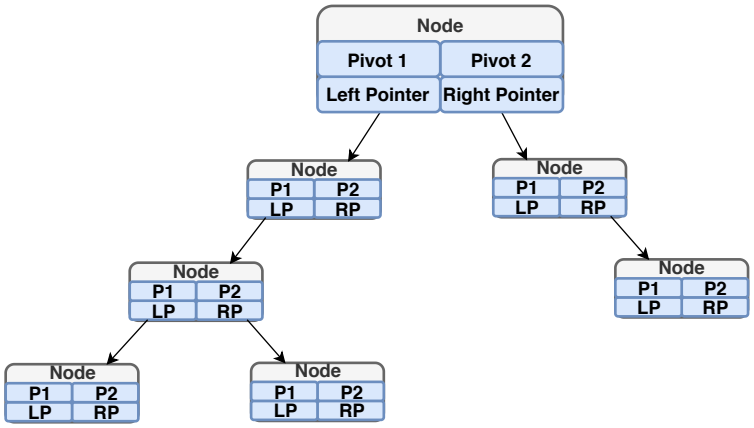


Figure 3.5: B-tree

3.3.3 Bagged Binary Tree (BB-tree)

The difference between our method and the previous one is that we added a "Bag" to the node elements shown in Fig. 3.6. This Bag contains a subset of data stored in it.

Algorithm 2 represents the construction recursive function of the BB-tree. In the first step, it inserts the objects to the first node's Bag until the cardinal limit of the Bag is reached. Then distribute the Bag to the left and right child nodes based on the distance between node's pivots (P_1, P_2) and the inserted object, that would create new nodes with their Bags. After transferring the Bag's data to child nodes we completely delete the Bag from the current node, that will lead us to having Bags at leaf nodes only, as you can see in Fig 3.7.

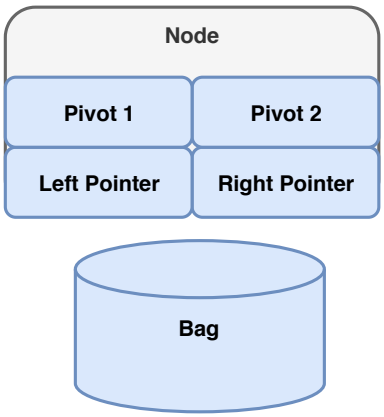


Figure 3.6: Node Elements

Algorithm 2 : Insertion (Tree, V_x)

```
1: if ( Tree.P1 = Null and Tree.P2 = Null) then
2:    if (length(Tree.Bag) < Bag.size) then
3:       Tree.Bag.append(V_x);
4:    else
5:       Create Left Node;
6:       Create Right Node;
7:       Extract Two Farthest Vectors (V1,V2);
8:       Tree.P1 = V1;
9:       Tree.P2 = V2;
10:      for  each vector in Tree.Bag do
11:         if Distance(vector, V1) < Distance(vector, V2) then
12:            Tree.Left.Bag.append(vector);
13:         else
14:            Tree.Right.Bag.append(vector);
15:         end if
16:      end for
17:      Delete(Tree.Bag)
18:      if  Distance(V_x, V2) < Distance(V_x, V2) then
19:         Tree.Left = Insertion(Tree.Left, V_x);
20:      else
21:         Tree.Right = Insertion(Tree.Right, V_x);
22:      end if
23:   end if
24: end if
25: return  Tree
```

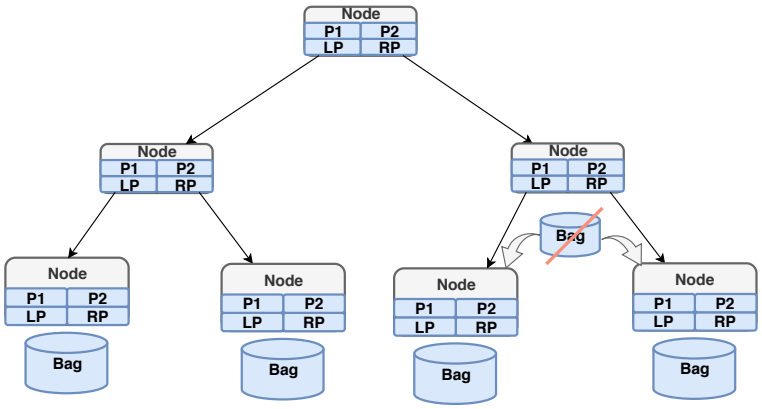


Figure 3.7: BB-tree construction

3.3.4 Distributed Bagged Binary Tree (DBB-tree)

This method is an extension of the previous one, and it consists using multiple machines connected in a network at the Cloud computing level. The first machine considered as server that distribute it's overloaded data to client machines. Every client machine considered as a server to other machines. Every machine create a local tree in it and linked to its clients with the leaf nodes of the local tree (see Fig 3.8). In the construction of the DBB-tree we needs both of algorithm 2 and 3 used in 4.

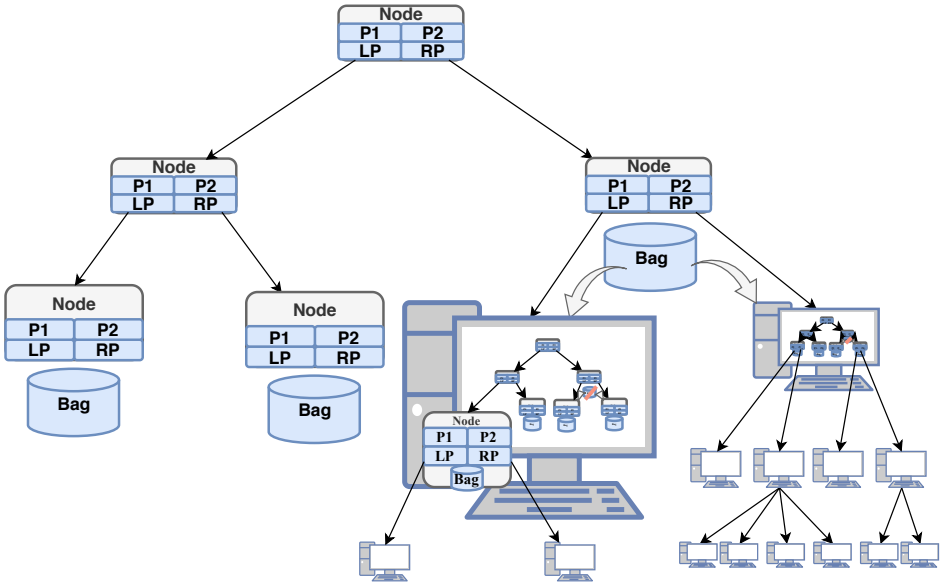


Figure 3.8: DBB-tree Construction

First step is to create the local tree of the server machine and fill it with data until it saturates using the insertion algorithm 2 explained in the previous subsection 3.3.3. The next inserted data after saturation will run through the tree as it going to be inserted in it, when it gets to the leaf node it creates two client machines and distribute the Bag's object to these machines then delete the Bag's content. The next coming data to the same path will fill the Bag until the machine is saturated a second time and again distribute it. Every machine will behave the same to its clients to create a global tree as shown Fig 3.8.

Algorithm 3 : InsertionSend (Tree, V_x)

```
1: if Tree is Not Leaf Node then
2:    if  Distance(V_x, Tree.P1) < Distance(V_x, Tree.P2) then
3:       Tree.Left = InsertionSend(Tree.Left, V_x);
4:    else
5:       Tree.Right = InsertionSend(Tree.Right, V_x);
6:    end if
7: else
8:    if (length(Tree.Bag) < Bag.size) then
9:       Tree.Bag.append(V_x);
10:   else
11:      Start(Right Machine,Left Machine);
12:      Extract Two Farthest Vectors from Tree.Bag (V1,V2);
13:      for each vector in Tree.Bag do
14:         if Distance(vector, V1) < Distance(vector, V2) then
15:            LeftMachine.send(vector);
16:         else
17:            RightMachine.send(vector);
18:         end if
19:      end for
20:      Delete(Tree.Bag)
21:      if  Distance(V_x, V1) < Distance(V_x, V2) then
22:         LeftMachine.send(V_x);
23:      else
24:         RightMachine.send(V_x);
25:      end if
26:   end if
27: end if
28: return  Tree
```

Algorithm 4 : Main ()

```
1: while Machine Not saturated do
2:     Tree=Insertion(Tree, V_x);
3: end while
4: while there is Data do
5:     Tree=InsertionSend(Tree, V_x);
6: end while
```

3.4 Search

The *k*NN search consists in searching the k nearest neighbors of each query point in the benchmark set given a specific distance (or similarity). Commonly, the *Euclidean* or the *Manhattan* distance is the most used ones. Figure 3.9 illustrates an example of *k*NN search with k = 4.

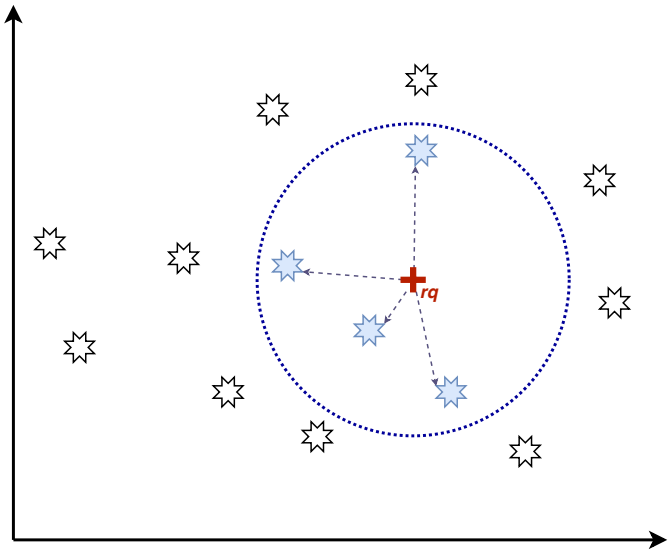


Figure 3.9: kNN search for k = 4.

Searching for objects in the first method (Subsection 3.3.1) would have an exponential complexity and the slowest time to find it, because we have to run through all data sequentially.

The search in the three last methods (Subsections 3.3.2, 3.3.3, 3.3.4) is done by calculating the distance between the query vector and the two pivots, while descending in the tree browsing the index and decreasing the query radius which corresponds to the distance to the k^e object in the ordered list A. The leaf nodes contain a subset of the indexed data with a maximum cardinal $c_{\max}$.
At the leaf level the algorithm is quite simple. In order to find the k closest neighbors of a leaf, we just need to sort them in an ascending order according distances to the query object. Then we return at most the first k objects already sorted.

Note, that this step does not really require sorting, but only a sequence of mergers. The complexity is "constant", in order of: $O(2.k)$ rather than: $O(2.k.\log_2 k)$. Relying on the decreases expressed by the computations of the radius is quite insufficient. It is necessary to estimate an upper-bound to the forthcoming k^{th} distance. In this way, the first call on the root node could be initialized with a much more satisfactory value than $+\infty$, the best estimation leading to a perfect search. Let us consider the case $c_{\max} = \sqrt{n}$, the time complexity is in the order : $O(\frac{1}{2}\sqrt{n}.log_2(k) + (log(n)/(\frac{1}{2}\sqrt{n}))k)$ with $n = |E|$.

Algorithm 5 kNN Search [52]

$$k\text{NN Search}\left(\begin{array}{l} N \in \mathcal{N}, \\ q \in \mathcal{O}, \\ k \in N^*, \\ d : \mathcal{O} \times \mathcal{O} \to R^+, \\ r_q \in R^+ = +\infty, \\ A \in (R^+ \times \mathcal{O})^N = \emptyset \end{array}\right) \in (R^+ \times \mathcal{O})^N$$

1: **if** $N = \perp$ **then**
2: **return** A
3: **else**
4: **let** $(p_1, p_2, r, N_p, n_e, n_r, N_f) = N$
5: **let** $d_1 = d(p_1, q)$
6: **let** $d_2 = d(p_2, q)$
7: **if** $|A| < k$ **then**
8: $r_q \leftarrow +\infty$
9: **else**
10: $(r_q, \cdot) \leftarrow A_k$
11: **end if**
12: **if** $d_1 < r_q$ **then**
13: $A \leftarrow$ k-Insert$(k, A, (d_1, p_1))$
14: **end if**
15: **if** $d_2 < r_q$ **then**
16: $A \leftarrow$ k-Insert$(k, A, (d_2, p_2))$
17: **end if**
18: **for** $i \in$ Ordre-visite(q, N) **do**
19: **if** $B(q, r_q) \cap N_i \neq \emptyset$ **then**
20: $A \leftarrow k$NN-BBC-index(N_i, q, k, d, r_q, A)
21: **end if**
22: **end for**
23: **return** A
24: **end if**

3.5 Conclusion

In this chapter, we presented a proposed indexing structure for Internet of Things data management at the Cloud Computing level. The main objective of this methods is to develop a new effective mechanism for storing, discovering and recovering data.

Chapter 4

Implementation and Results

4.1 Introduction

Since the objective of our structures is to build an index and respond effectively to kNN queries, in this chapter we present an experimental evaluation of our structures, on real IoT data for indexing, and query processing in WSANs (wireless sensor and actuator network). We have launched several experiments on our three structures: Binary Tree (B-tree); Bagged Binary Tree (BB-tree); Distributed Bagged Binary Tree (DBB-tree). See Table 4.1) to demonstrate the wide range of applicability of our index.

4.2 Data set

In these experiments, we used the following three datasets and Table 4.1 shows some characteristics of these datasets.

- DB1 : The first dataset is GPS Trajectory [53] which contains the trajectories of GO!Track application users using a variety of means of transport in north-east Brazil [54].
- DB2 : The second dataset contains a list of locations and other places with their geographical coordinates obtained from the BD-L-TC topographic database. [55]
- DB3 : The third dataset represents a set of moving object feature vectors obtained

by an object tracking simulator using wireless cameras in the wireless multimedia sensor network during a random simulation.

Dataset	Objects number	Dimensions
Geographical Coordinates	988	2
GPS Trajectory	18107	3
Tracking Data	62702	20

Table 4.1: Characteristics of the dataset

4.3 Simulation Setup:

We used **Google Colaboratory** platform which is a free research project by Google. It's a Jupyter notebook environment that requires no setup and runs entirely on the cloud, and it can be synchronized live with Google Drive. It is a Python based code and provides the Tesla GPU acceleration.

Technical characteristics of the used setup are shown below:

- CPU: 1 single core hyper threaded (1 core, 2 threads) Xeon processors 3GHz (No turbo boost) 45MB cache
- GPU: 1xTesla K80 Compute 3.7 2496 CUDA cores 12GB GDDR5 VRAM
- RAM: 12.6 GB
- Disk: 33 GB

4.4 Measures relating to the quality of the index structure

The study on our structure with statistics on the quality of the index is very important for the search algorithm. It can be determined by various measures, such as index balancing,

node fill rate and object distribution in the index.

Therefore, our goal is to verify that the tree built is relatively well balanced. Note that our proposal is a « paginated » tree (*Bucketed Tree*), i.e. where leaf nodes are made up of a set of elements instead of a single one.

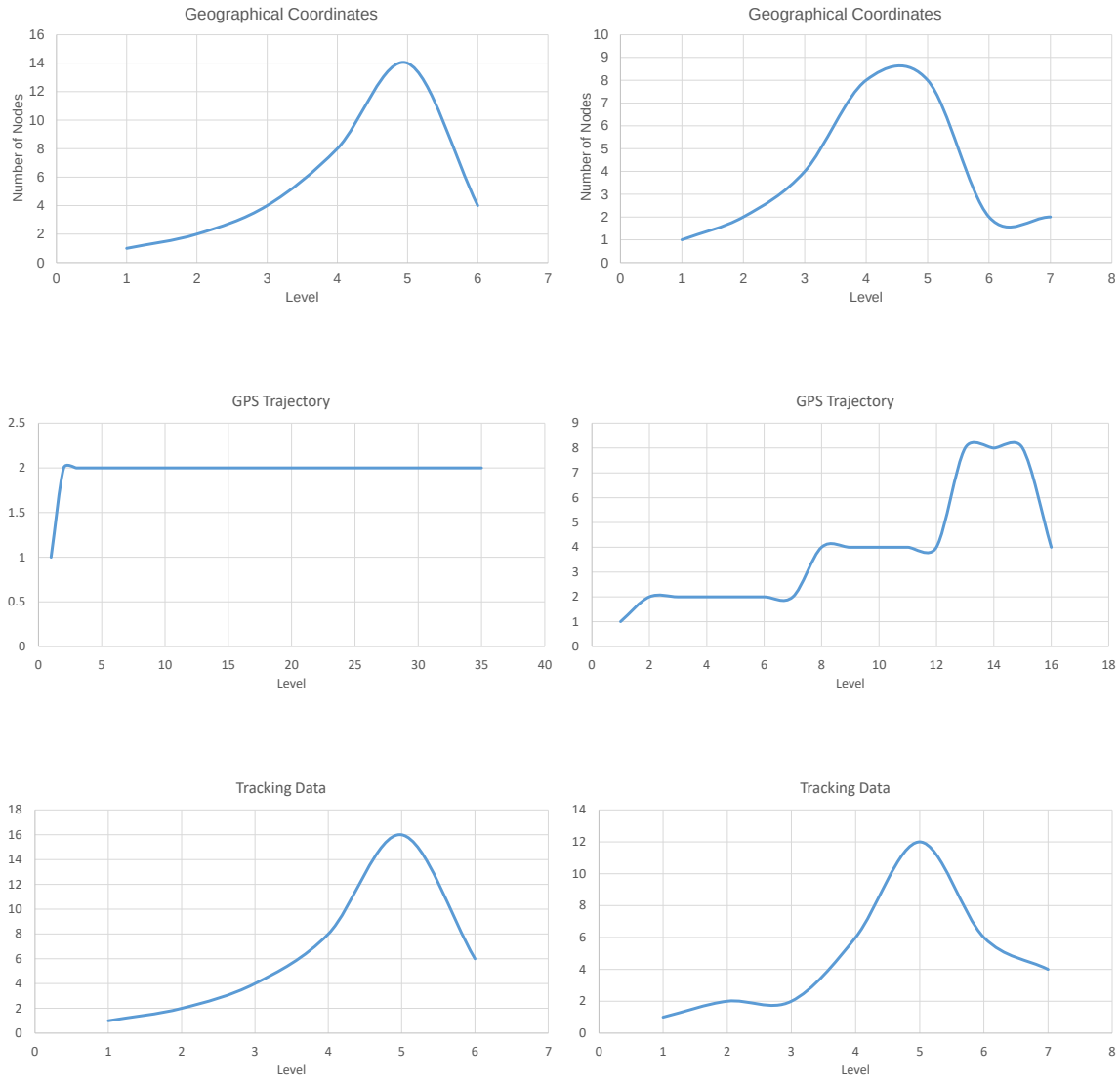
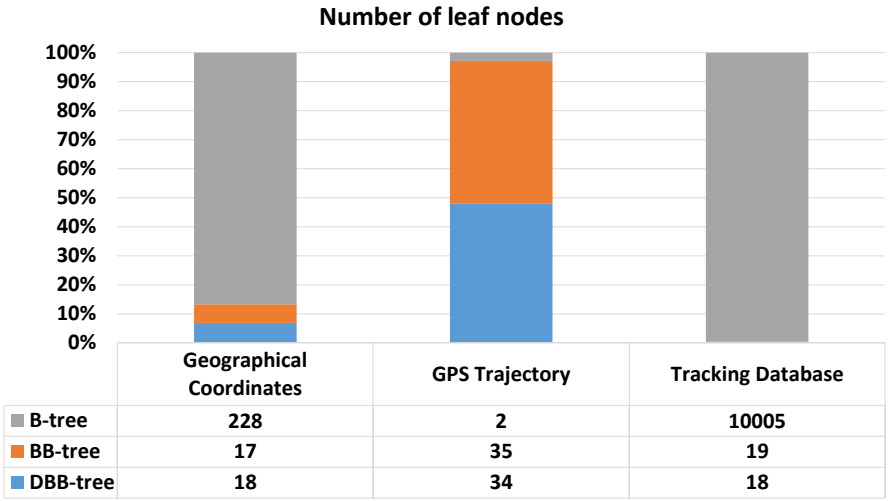


Figure 4.1: Number of nodes per level in BB-tree and DBB-tree

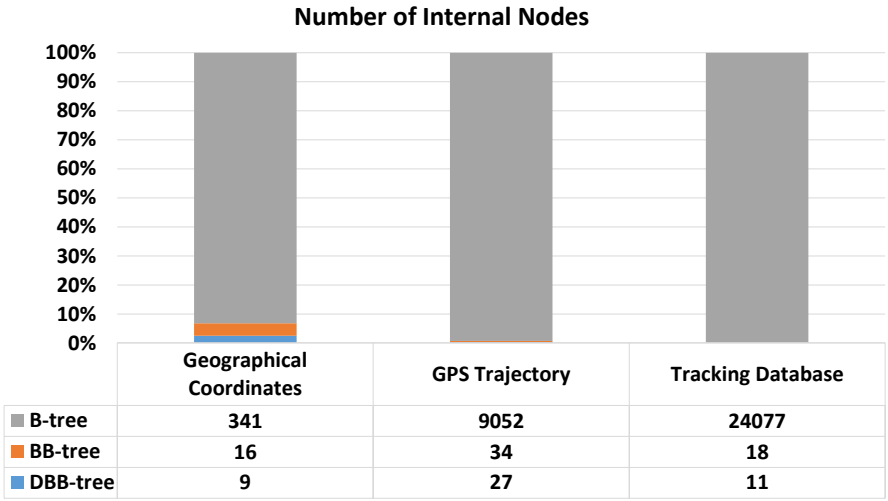
The graphics in Figure 4.1 show the number of nodes present at each level of the BB-tree and DBB-tree versions as well as the maximum depth reached. The numerical details of these histograms are also reported. In general, the structure ensures a quasi-balancing of the indexes, so the problem of degeneration does not arise in this case. As exception,

for GPS Trajectory on BB-tree structure, we observed a risk of degeneration, while the quality has been ameliorated on DBB-tree.



	Geographical Coordinates	GPS Trajectory	Tracking Database
B-tree	228	2	10005
BB-tree	17	35	19
DBB-tree	18	34	18

Figure 4.2: Number of leaf nodes statistics.



	Geographical Coordinates	GPS Trajectory	Tracking Database
B-tree	341	9052	24077
BB-tree	16	34	18
DBB-tree	9	27	11

Figure 4.3: Number of internal nodes statistics.

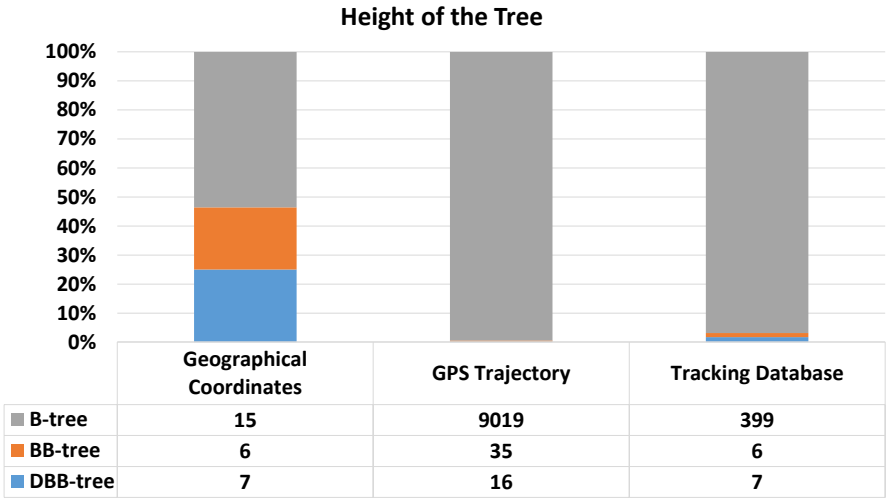


Figure 4.4: Height of trees statistics.

Next, we examined to the distribution of objects into the index and the filling rate of the leaf nodes (see 4.2 4.3 4.4). Of course, we worked on the same collections and the same parameters according to the results found previously. We calculated the average number of leaf nodes, internal nodes and the height of the tree during the construction process in all levels of the index. We observed that in B-tree the number of leaf nodes is too high comparing to BB and DBB trees because the leaf nodes are made up of a set of elements (bags) instead of a single one, except the GPS trajectory dataset.

	Left objects number	Right objects number	Neither left nor right
Master	76	104	
Client 1	68	45	
Client 2	126	64	
Client 3			89
Client 4	67	47	
Client 5			95
Client 6	51	57	
Client 7			38
Client 8			65

Table 4.2: Distribution of objects in the index "Geographical Coordinates"

	Left objects number	Right objects number	Neither left nor right
Master	2396	696	
Client 1	2001	987	
Client 2	2003	14	
Client 3	2458	500	
Client 4	1525	522	
Client 5	2074	523	
Client 6	1489	919	

Table 4.3: Distribution of objects in the index "GPS Trajectory"

	Left objects number	**Right objects number**	**Neither left nor right**
Master	2326	364	
Client 1	3076	6193	
Client 2	4641	6094	
Client 3	4190	4300	
Client 4	3616	2897	
Client 5	2844	3851	
Client 6	5663	2645	
Client 7			1753
Client 8			3248
Client 9			3077
Client 10			1924

Table 4.4: Distribution of objects in the index "Tracking Data"

Afterwards, the tables 4.2, 4.3 and 4.4 shows the number of objects for each region of the index for each collection. Since we used the construction algorithms for the DBB-tree version, we cannot reasonably expect that each sheet will contain exactly $c_{\max}$ objects; we cannot imagine that the balance will be perfect, In this context, the observed distribution is then very satisfactory. considering the "neither left nor right" column for the first and the last datasets there are some client machines with one node created that have neither left nor right nodes, because they didn't reach the maximal bag's size.

4.5 Measures relating to the cost of the construction algorithms

	Number of Distances			**Number of Comparisons**		
	DBB-tree	*BB-tree*	*B-tree*	*DBB-tree*	*BB-tree*	*B-tree*
Geographical Coordinates	91336	87420	16770	45668	43710	8385
GPS Trajectory	26354318	17642840	163149008	13177159	8821420	81574504
Tracking Data	312928804	225491532	19402440	156464402	112745766	9701220

Table 4.5: Measurements of tree construction

In this part 4.5 we launched the experiments on the three prototypes under the following conditions: In each of the scenarios, we calculate:

- The number of distances calculated;
- The number of comparisons required;

We observed that number of the distances and comparisons is less in B-tree than BB-tree and DBB-tree. The explanation of this is that calculating the furthest points of the bag, to choose the pivots of the node needed more distances and comparison calculations.

	Geographical Coordinates	GPS Trajectory	Tracking Data
Master	0.000566	0.005553	0.011763
Client 1	0.000626	0.004926	0.038746
Client 2	0.001664	0.003396	0.045461
Client 3	0.000662	0.005157	0.043167
Client 4	0.000904	0.004483	0.027769
Client 5	0.000613	0.004397	0.026856
Client 6	0.001018	0.004026	0.036133
Client 7	0.000399		0.007619
Client 8	0.000464		0.010049
Client 9			0.017279
Client 10			0.007991
Total	0.002231	0.01071	0.057225

Table 4.6: DBB-tree 5NN search time (s)

	Geographical Coordinates	GPS Trajectory	Tracking Data
BB-tree	0.001870	0.038222	0.289234
DBB-tree	0.002231	0.01071	0.057225

Table 4.7: Statistics of DBB and BB trees for 5NN search time(s).

In the following table 4.5, we can clearly see that the number of objects and dimensionality have a direct influence on the search time in our index. In eleven real cloud machines, our indexes show that the search time is almost balanced between them, not surprisingly, the structure is so quasi-balanced which reflects on the search algorithm. In the following table 4.5, we observe that we increase the number of objects our structure proves this performance. And the DBB distributed structure remains efficient.

4.6 Conclusion

In this chapter, we implemented and tested our structures on three different real IOT collections and we evaluated our construction and search KNN algorithms. We have

also launched several experiments on the three prototypes of our structure: Binary Tree (B-tree); Bagged Binary Tree; Distributed Bagged Binary Tree.

General Conclusion

The amount of information digitized has increased considerably in recent years, the growing innovations in **I**nformation and **C**ommunication **T**echnologies (ICT) in the computer and communications industry have enabled distributed systems researchers to design new systems and models to meet the needs of an increasingly diverse user community. Therefore, the main objective of our work has been to have an efficient search system in these data masses.

We studied a set of indexing techniques in spatial metrics with a strict subset. We had neither the time nor the space to make an exhaustive overview. As a result, we were able to highlight only the metric space indices. Based on what appear to be the best recent approaches, we proposed a distributed approach to indexing data in a metric space, essentially inspired by the binary tree. In summary, our proposal can be considered as a paginated and parameterized version of the latter.

Nevertheless, the problem encountered each time by these techniques is that of the "curse of the dimension". In a second step, we developed a distributed version to improve performance, which, in our opinion, should stagnate, or at least progress in this family of indices.

The perspectives we can generate from this work are expressed in short works and ideas of a more distant scope. First of all, it seems useful to continue the work in order to: - Propose a synchronization algorithm to minimize the number of client machines during the search algorithm; - Develop a real prototype allowing a real capture of IoT data, and process this data in real time. - Find a compromise between the size of the bags and the number of customer machines. - An extremely useful approach when processing complex data in large quantities is to combine several indexes, using different metrics, rather than having to create a new index.

Bibliography

[1] "The ierc iot definition," http://www.internet-of-things-research.eu/about$_i ot.htm, accessed : 2019 - 07 - 10.

[2] "Types of cloud computing - advantages and disadvantages!" https://convergenceservices.in/blog/corporate-blog/436-public-private-and-hybrid-cloud-computing-advantages-and-disadvantages.html, accessed: 2019-07-10.

[3] Y. Fathy, P. Barnaghi, and R. Tafazolli, "Large-scale indexing, discovery, and ranking for the internet of things (iot)," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, p. 29, 2018.

[4] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[5] "Internet des objets esie consulting," https://www.esie-consulting.fr/notre-crm/internet-des-objets/, accessed: 2019-07-10.

[6] S. Rottenberg, "Modèles, méthodes et outils pour les systèmes répartis multiéchelles," Ph.D. dissertation, Institut National des Télécommunications, 2015.

[7] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in *2010 39th International Conference on Parallel Processing Workshops*. IEEE, 2010, pp. 275–279.

[8] D. Uckelmann, M. Harrison, and F. Michahelles, "An architectural approach towards the future internet of things," in *Architecting the internet of things*. Springer, 2011, pp. 1–24.

[9] B. Christophe, C. Damien, G. Thomas, J. Julia, L. Renaud, and N. Nha-Khanh, "La sécurité de l'internet des objets," 2017.

[10] P.-J. Benghozi, S. Bureau, and F. Massit-Folea, "L'internet des objets. quels enjeux pour les européens?" 2008.

[11] H. Duce, "Internet of things in 2020," 2008.

[12] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[13] L. Tan and N. Wang, "Future internet: The internet of things," vol. 5, 09 2010, pp. V5–376.

[14] G. Gan, Z. Lu, and J. Jiang, "Internet of things security analysis," in *2011 International Conference on Internet Technology and Applications*, Aug 2011, pp. 1–4.

[15] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of internet of things," vol. 5, 09 2010, pp. V5–484.

[16] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things," *Future Gener. Comput. Syst.*, vol. 56, no. C, pp. 684–700, Mar. 2016. [Online]. Available: https://doi.org/10.1016/j.future.2015.09.021

[17] "PrÉparer la rÉvolution de l'internet des objets," https://www.arcep.fr/uploads/tx$_g$*spublication/livre$_b$lanc$_I$oT* − 01 − *cartographie* − 071116.*pdf*?*wb*48617274 = 1*C*436397, *accessed* : 2019 − 07 − 10.

[18] "Les différentes couches d'une infrastructure iot," https://www.silicon.fr/hub/hpe-intel-hub/les-differentes-couches-dune-infrastructure-iot, accessed: 2019-07-10.

[19] S. Naveen, "Study of iot: Understanding iot architecture, applications, issues and challenges," 05 2016.

[20] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: The internet of things architecture, possible applications and key challenges," in *2012 10th International Conference on Frontiers of Information Technology*, Dec 2012, pp. 257–260.

[21] "Utility computing," https://en.wikipedia.org/wiki/Utility$_c$*omputing*, *accessed* : 2019 − 07 − 10.

[22] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100

[23] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology (NIST), Gaithersburg, MD, Tech. Rep. 800-145, September 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[24] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010. [Online]. Available: https://doi.org/10.1007/s13174-010-0007-6

[25] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *2010 International Conference on Intelligent Computing and Cognitive Informatics*, June 2010, pp. 380–383.

[26] "Study on advantages and disadvantages of cloud computing," https://pdfs.semanticscholar.org/ada5/876e216130cdd7ad6e44539849049dd2de39.pdf, accessed: 2019-07-10.

[27] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys*, vol. 30, no. 2, pp. 170–231, 1998.

[28] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Computing Surveys*, vol. 33, no. 3, pp. 322–373, Sep. 2001.

[29] H. Samet, *Foundations of Multidimensional And Metric Data Structures*. Morgan-Kaufmann, Sep. 2006, 993 p.

[30] X. L. C. S. J. L. Chen, Y. Gao and G. Chen, "Efficient metric indexing for similarity search and similarity joins," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1041-4347. IEEE, 2017, pp. 556 – 571.

[31] A. S. Gonzaga and R. L. F. Cordeiro, "A new division operator to handle complex objects in very large relational datasets." in *EDBT*, 2017.

[32] W. Y. Wan Yuchai, Liu Xiabi, "Cd-tree: A clustering-based dynamic indexing and retrieval approach," *Intelligent Data Analysis*, vol. 21, pp. 243–261, 2017.

[33] P. Bolettieri, F. Falchi, C. Lucchese, Y. Mass, R. Perego, F. Rabitti, and M. Shmueli-Scheuer, "Searching 100m images by content similarity," in *Post-proceedings of the 5th Italian Research Conference on Digital Library Systems (IRCD)*, Padova, Italy, Jan. 2009, pp. 88–99.

[34] M. Batko, D. Novak, F. Falchi, and P. Zezula, "On scalability of the similarity search in the world of peers," in *Proceedings of the 1st international conference on Scalable information systems (InfoScale)*. Hong Kong, China: ACM Press, May 2006, pp. 20–31.

[35] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," *Proceedings of the 23rd VLDB International Conference*, pp. 426–435, 1997.

[36] C. Traina Jr, A. Traina, B. Seeger, and C. Faloutsos, "Slim-trees: High performance metric trees minimizing overlap between nodes," *International Conference on Extending Database Technology (EDBT)*, 2000.

[37] R. d. S. T. N. J. L. Jurandy Almeida, Eduardo Valle, "Dahc-tree: An effective index for approximate search in high-dimensional metric spaces," *Journal of Information and Data Management*, vol. 1, no. 3, pp. 375–390, october 2010.

[38] I. R. V. Pola, A. J. M. Traina, C. Traina, and D. S. Kaster, "Improving metric access methods with bucket files," in *Similarity Search and Applications*, G. Amato, R. Connor, F. Falchi, and C. Gennaro, Eds. Cham: Springer International Publishing, 2015, pp. 65–76.

[39] G. Navarro, "Searching in metric spaces by spatial approximation," in *In Proceedings of String Processing and Information Retrieval (SPIRE99), Cancun, Mexico*, 1999.

[40] ——, "Searching in metric spaces by spatial approximation," *The VLDB Journal*, 2002.

[41] D. Arroyuelo, "A dynamic pivoting algorithm based on spatial approximation indexes," in *Similarity Search and Applications - 7th International Conference, SISAP 2014, Los Cabos, Mexico, October 29-31, 2014. Proceedings*, 2014.

[42] R. Pagh, F. Silvestri, J. Sivertsen, and M. Skala, "Approximate furthest neighbor in high dimensions," in *Similarity Search and Applications - 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings*, 2015.

[43] B. Z. C. S. J. H. Y. K. Y. Lu Chen, Yunjun Gao, "Pivot-based metric indexing," in *Proceedings of the VLDB Endowment*, 2016.

[44] L. Chen, Y. Gao, X. Li, C. S. Jensen, and G. Chen, "Efficient metric indexing for similarity search and similarity joins," *IEEE Transactions on Knowledge & Data Engineering*, vol. 29, no. 3, pp. 556–571, mar 2017. [Online]. Available: doi.ieeecomputersociety.org/10.1109/TKDE.2015.2506556

[45] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," *proceedings of the 4th Annual In ACM-SIAM Symposium on Discrete Algorithms*, pp. 311–321, 1993.

[46] F. Nielsen, "Bregman vantage point trees for efficient nearest neighbor queries," in *Proceedings of Multimedia and Exp (ICME). IEEE*, 2009.

[47] F. A. W. chee Polly Mei-shuen Chan Yin-Ling Cheung Yiu Sang Moon, "Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances," *The VLDB Journal Very Large Data Bases*, 2012.

[48] T. Bozkaya and M. Özsoyoglu, "Indexing large metric spaces for similarity search queries," *ACM Transactions on Database Systems*, vol. 24, pp. 361–404, Sep. 1999.

[49] R. R. Curtin, "Faster dual-tree traversal for nearest neighbor search," in *Similarity Search and Applications - 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings*, 2015.

[50] M. Zierenberg and I. Schmitt, "Optimizing the distance computation order of multi-feature similarity search indexing," in *Similarity Search and Applications - 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings*, 2015.

[51] I. R. V. Pola, C. Traina, and A. J. M. Traina, "The nobh-tree: Improving in-memory metric access methods by using metric hyperplanes with non-overlapping nodes," *Data & Knowledge Engineering*, vol. 94, no. PA, pp. 65–88, nov 2014. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2014.09.001

[52] Z. Kouahla, "Indexation dans les espaces métriques Index arborescent et parallélisation," Theses, Université de Nantes, Feb. 2013. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00912743

[53] "Gps trajectories data set," https://archive.ics.uci.edu/ml/datasets/GPS+Trajectories, accessed: 2019-07-10.

[54] M. O. Cruz, H. Macedo, and A. Guimarães, "Grouping similar trajectories for carpooling purposes," in *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, Nov 2015, pp. 234–239.

[55] "Geographical coordinates," https://data.public.lu/fr/datasets/r/a7d551d7-f374-491a-ab93-63715b98e6dd, accessed: 2019-07-10.