Distributed Trajectory Database Design Based on Massively Parallel Processing (MPP) Architecture

Gong Sicong
5711932
Delft University of Technology
1st supervisor: Dr.Ir. Martijn Meijers
2nd supervisor: Drs. Wilko Quak

January 10, 2024

Contents

1 Introduction

In this section, the three driving forces, including the supply side, demand side and technology side of managing trajectory data are first presented. Then, the three criteria which would determine a good implementation outcome are also discussed.

1.1 Background

With the update of the mobile Internet of Things (IoT) infrastructure and the popularization of corresponding devices, massive amounts of various spatio-temporal data are emerging, collected and stored, highlighting the arrival of the Big Data era in the field of geomatics (Li (2019); Li et al. (2020)). Among them, trajectory data (shown in Figure 1) of moving objects (such as vehicles, humans and animals) collected by the Global Navigation Satellite System (GNSS), Global System for Mobile Communications (GSM) etc., is uniquely valuable because of the large amount of information it contains.

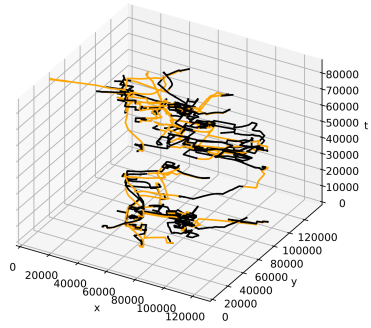

Figure 1 General Definition of Trajectory: a trajectory is a sequence of 3D/4D points in 3D(X, Y, T)/4D(X, Y, Z, T) space, each point may have some attributes as semantics (Alsahfi et al. (2020)).

Although the storage burden can be solved by horizontally extending inexpensive storage media, the utilization and reproduction of trajectory data is still limited by its large size and complex structure. This seriously hinders the added-value extraction from historical as well as real-time data through data analysis or mining technologies in research, engineering and decision-making, particularly in real-time applications.

To cope with the query, computation and analysis of massive data, distributed/parallel storage and computation are gradually replacing traditional centralized/serial strategies, with Apache Hadoop and Massively Parallel Processing (MPP) ecosystems as the dominant ones. While Hadoop is suitable for high-throughput applications, MPP is more suitable for low-latency applications.

Although these technologies can be well applied in industries such as the Internet and Finance, actions to expand them to the field of geomatics are still progressing. Considering the characteristics of trajectory data (non-structural, dynamically evolving,

multi-dimensional etc.), it is still necessary to reconsider multiple aspects such as data modelling, clustering, indexing, and data distribution etc.

1.2 Motivation

This research is meant to explore the possibilities (alternatives) to extend the distributed technologies for the trajectory data. Specifically, this research will be motivated to reduce the pressure of data storage, accelerate the speed of data queries, and balance the space and time complexity, as well as the complexity of thinking and coding.

Time Aspect: The main research focus is on enhancing the query (especially spatio-temporal related ones such as selection by intersection) speed, meeting the client's demands.

Storage Aspect: Though not the primary focus, the research aims to minimize the redundancy in the original data, exploring compression strategies for efficient storage.

Complexity Aspect: Space and time complexity are often mutually exclusive, for instance, intricate compression algorithms may sacrifice speed for higher compression ratios. Also, fine design would improve performance but increase the development cost which is not realistic for this research.

2 Related Work

In this section, relevant works from four aspects are presented and discussed. Specifically, they are how to model the trajectory phenomenon into the database, what are the strategies to speed up the accessing process, and the typical storage and computation architecture, as well as the typical applications.

2.1 Trajectory Modelling

Depending on the aggregation level and whether structured, the trajectories are typically modelled as points, grids and sequences (Ribeiro de Almeida et al. (2020)).

Point-Based Modelling: It takes the individual points (shown in Figure 2 (a)) as the basic storage element. Besides using plain tables, considering X, Y, Z, T and other attributes as homogenous dimensions, some separate organizing dimensions and property dimensions. While organizing dimensions are usually encoded somehow for clustering a indexing, the property dimensions are kept as usual (De Vreede (2016); Meijers et al. (2016); Meijers and van Oosterom (2018); li2 (2020); Liu (2022)).

Grid-Based Modelling: It takes the regular cell/cube (shown in Figure 2 (c)) as the basic storage element. Pre-calculation/aggregation of certain semantics at certain cells in the regular spatial-temporal cubes is needed. This kind of modelling is especially

good for OLAP applications due to its regular multi-dimensional nature (Leonardi et al. (2010, 2014)).

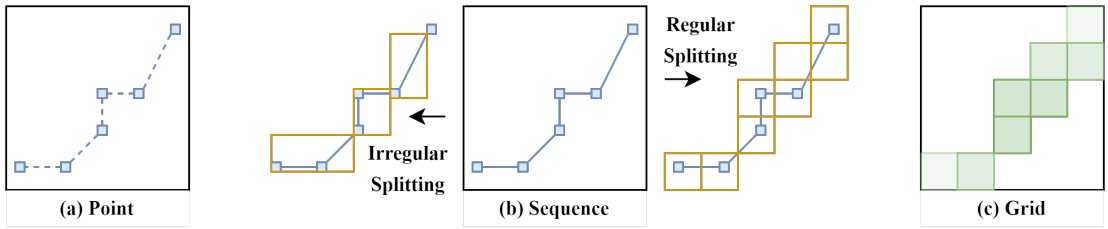

Figure 2 Three Typical Trajectory Models: individual points with implicit lineString geometries are stored in the point-based modelling; explicit geometries are stored in the sequence-based modelling; aggregated values (such as average speed in this example) are stored in the grid-base modelling.

Sequence-Based Modelling: It takes the vector-based lineString (shown in Figure 2 (b)) as the basic storage element after reconstructing (geometrical/semantical) meaningful sequences of points (Pelekis et al. (2015); Zimányi et al. (2020); Biljecki et al. (2013); Pfoser et al. (2000)). The semantics are assigned to each sequence as attributes. Depending on the application, sequences are often modelled in 3D or 4D space.

The grid-based modelling has the coarsest granularity, good at visualization, however, the geometries are lost during the aggregation process, making geometrical operations impossible. On the contrary, point-based modelling has the finest granularity and most flexibility, however, the geometries are not explicitly reconstructed, leading to bad cases (shown in Figure 3) when doing certain operations.

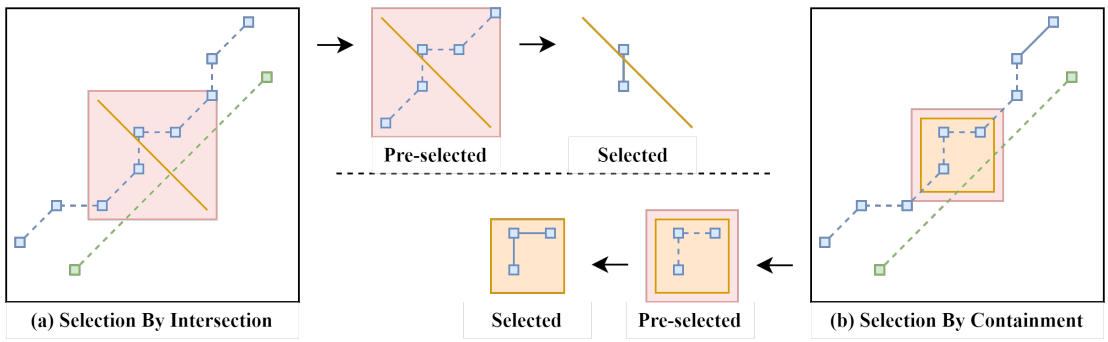

Figure 3 Bad Cases of Point-Based Modelling: when doing intersection (orange line at left) or containment selection (orange box at right), it has to first select points inside a certain range and do reconstruction since there are no explicit geometries. How to determine the pre-selected range is difficult, especially when the sampling rates are different. The green trajectory intersects the orange line or is contained inside the orange box, but is missed when doing selection as its points are not pre-selected.

The sequence-based modelling is a compromise/balance between point-based modelling and grid-based modelling, aggregating at certain levels (reducing redundancy)

while keeping the geometries. The main disadvantage is the unstructured nature such as different sampling rates or lengths for each trajectory. Thus, the core when doing sequence-based modelling is how to transform the unstructured trajectories into structured ones.

After the reconstruction of the whole trajectory, further splitting is needed which is a preparation for spatial accessing methods. Typically, there are irregular splitting and regular splitting (shown in Figure 2 (b)). The irregular method splits the trajectory geometrically (such as by truing points) or semantically (such as by driving state). The regular method splits the trajectory by regular spatio-temporal cubes. A combination of the two splitting methods is also possible (shown in Figure 4).

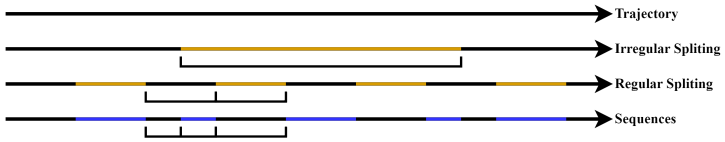

Figure 4 Combination of Two Splittings: the trajectory is first irregularly split by state (such as whether there are passengers) and then regularly split. In this way, the length of each sequence can be controlled, and the sequence can be contained in the cells to facilitate subsequent clustering and indexing.

2.2 Trajectory Accessing

Accessing methods are used to speed up the CURD operations in the DBMS. B-tree, BRIN etc. are traditional non-spatial indexing methods. However, trajectory by nature is a spatially related phenomenon with scale and neighbouring effects, when storing the trajectory, there's a need to consider spatial locality, homogeneity, regularity etc.

Spatial Clustering and Grouping: It means using spatial locality to store the closer elements in the real world (or user-defined space) also closer in the storage media which organizes data in a liner way. The locality is usually application-dependent and specified by clients. There are several ways to define the closeness such as Space-Filling Curves (van Oosterom and Vijlbrief (1996)), and some machine learning algorithms such as K-Means could be used, too.

Dynamically Balanced Search Tree: It exemplified by R-tree (shown in Figure 5 (a)) and its variants (R*tree, R+tree, Hilbert R-tree, etc.), dynamically consider the distribution of objects during dimensional space partitioning. The primary advantage lies in achieving a balanced index structure, flexiblely maintaining objects within each node (both leaf and intermediate nodes), thereby contributing to retrieval performance (van Oosterom (1999); Liu (2022); Guan (2020); Mahmood et al. (2019)).

However, the construction and updating of this balanced tree index are intricate, posing challenges for practical deployment, especially in distributed environments. As the data volume increases, the depth of the index tree grows, resulting in a rapid decline in retrieval efficiency. Dynamically Balanced Search Trees are well-suited for scenarios

where the spatial distribution of objects is uneven, but their complexity in construction and updates make them less suitable for distributed environments or applications with rapidly increasing data volumes.

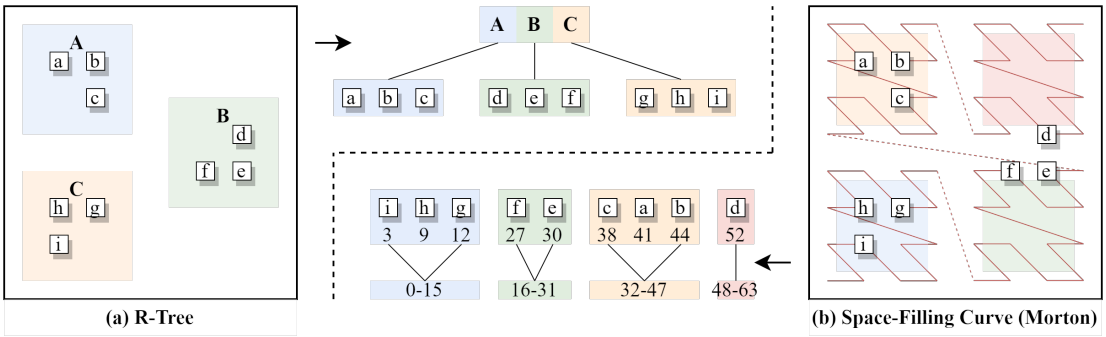

Figure 5 Two Typical Spatial Accessing Methods

Regular Dimensional Space Tessellation: It is represented by indices like the grid index and space-filling curve (shown in Figure 5 (b)), involving static regular segmentation of dimensional space. The primary advantage lies in its simplicity of construction and maintenance, eliminating the need for adjustments when adding new data and ensuring ease of use.

However, this approach struggles with the adaptive handling of data distribution. Some index division units may become too dense, while others may remain devoid of data, resulting in unstable retrieval efficiency. Regular Dimensional Space Tessellation is well-suited for scenarios prioritizing easy construction and maintenance, particularly in environments where data distribution remains relatively stable, and frequent adjustments to the index structure are undesirable.

In seeking a compromise between the two main steam accessing methods, grid division index incorporating spatial distribution, and multi-level indexing hybridizing multiple strategies emerge as potential solutions. It becomes evident that no single method universally suits all scenarios. The optimization for specific refinement and complexity may inadvertently result in diminished performance in alternative applications.

Notably, while regular methods may excel in distributed environments, each approach presents trade-offs, emphasizing the importance of selection based on the specific demands and characteristics of the given data. In conclusion, the diverse nature of spatial indexing methodologies underscores the need for a understanding and careful consideration of trade-offs to ensure optimal performance in varied scenarios.

2.3 Trajectory Organizing

When jumping from the data storage and computing in the geomatics field back to a broader scope, there's a need to care about the macro classification of the DBMS architectures (Elmasri et al. (2015); Wang (2022); Guan (2020)).

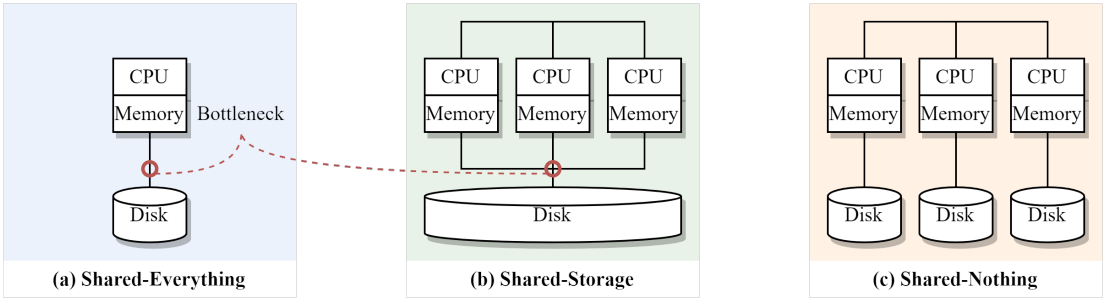

Figure 6 Three Typical DBMS Architecture: the performance bottleneck of shared-everything and shared-storage architecture mainly comes from the IO (data exchange between disk and memory)

Shared-Everything: In this architecture, CPU, Memory, and Disk resources are shared (shown in Figure 6 (a)). Depending on the application, the bottleneck may occur at the CPU cache or IO, causing a decline in performance.

Shared-Storage: In this architecture, disk resources are shared (shown in Figure 6 (b)). Vertical scaling is possible (though expensive), and the bottleneck may still be associated with IO operations.

Shared-Nothing: In this architecture, CPU, Memory, and Disks are all distributed (shown in Figure 6 (c)). Horizontal scaling helps alleviate the bottlenecks mentioned in the previous two models, but it may introduce additional costs and marginal effects. There are two main stream distributed system architectures, MPP and Hadoop. Their comparison with traditional databases is shown in Table 1.

Table 1 Architecture Comparison Adopted from Wang (2022)

Feature	Traditional Database	Hadoop	MPP Database
Volume	GB-TB	PB-EB	TB-PB
Robustness	High	High	Medium
Scalability	Low	High	Medium
Delay	Medium	High	Low
Throughput	Low	High	Medium
Data Type	Structured	All	Structured

1. MPP (Massively Parallel Processing) Ecosystem: MPP ecosystems excel in high-performance data processing through a shared-nothing DBMS-based architecture, distributing computational tasks across multiple nodes for efficient handling of large datasets. Horizontal scalability enhances its suitability for complex analytics and data warehousing applications.
2. Hadoop Ecosystem: The Hadoop ecosystem, anchored by the Apache Hadoop core, embraces distributed computing for large-scale data processing. With a shared-nothing architecture, Hadoop facilitates parallel computation and fault tolerance, making it a versatile and scalable solution for diverse big data needs.

Some variants such as Spark convert disk-based calculations into memory-based calculations, reducing latency.

In a distributed environment, it is better to store objects that are closer also closer in the storage media, however, in the distributed system, this may not be the case (shown in Figure 7).

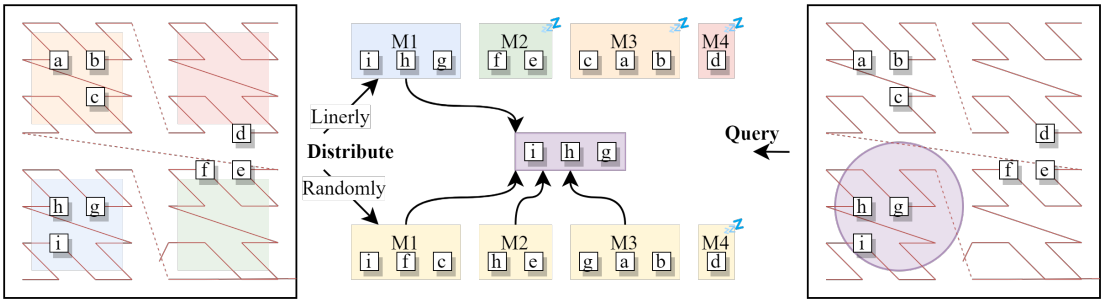

Figure 7 Different Distributing Strategies: If data is distributed linearly and orderly according to the Space-Filling Curve, closer data will be stored on the same machine. When only relevant data is queried, only one machine is working, and the other machines are "sleeping".

2.4 Trajectory Application

There are plenty of applications, including selection, matching, bundling etc. Some of these queries and computations could be sped up by certain strategies.

Selection: It is the most basic operation and serves as the basis to support other more complex and advanced applications.

1. Selection by Intersection: select the features that intersect with the querying features (such as lineString or polygon).
2. Selection by Containment: select the features that are contained inside the querying features (such as polyhedron or sphere). It is also called range selection, depending on the accessing methods, the querying process could be sped up such as the Sweep algorithm (shown in Figure 8) (Liu (2022)).
3. Selection by Identifiers: select the features that belong to one object, in the taxi trajectory case, it is selected by the unique identifier of the vehicle.

Matching: It can complete applications independently (machining two trajectories or matching the trajectories with roads) and can also be used as a prerequisite for other applications (measuring the similarity/distance between trajectories used for clustering/bundling).

Bundling: It mainly works for visualization (shown in Figure 9). The use of bundling remedies visual clutter and reveals high-level patterns (Holten and Van Wijk (2009)).

Note that some applications may need to be adjusted due to the distributed architecture.

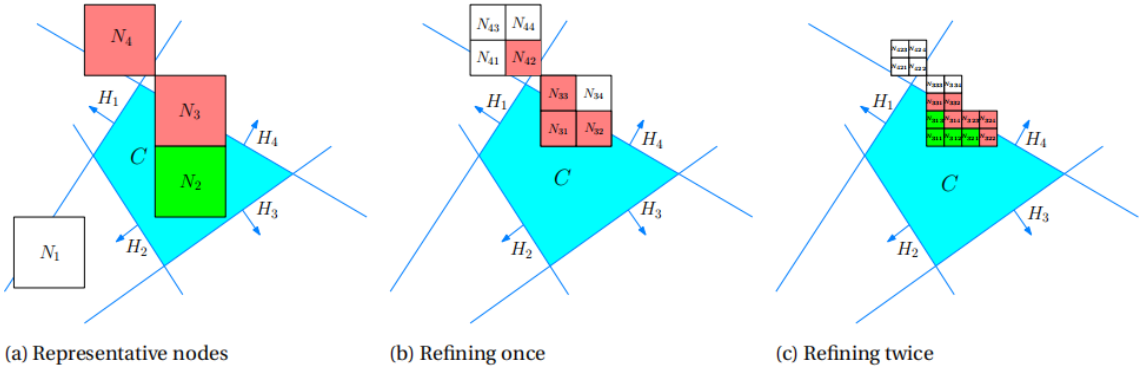

Figure 8 Sweep Algorithm Adopted from Liu (2022).

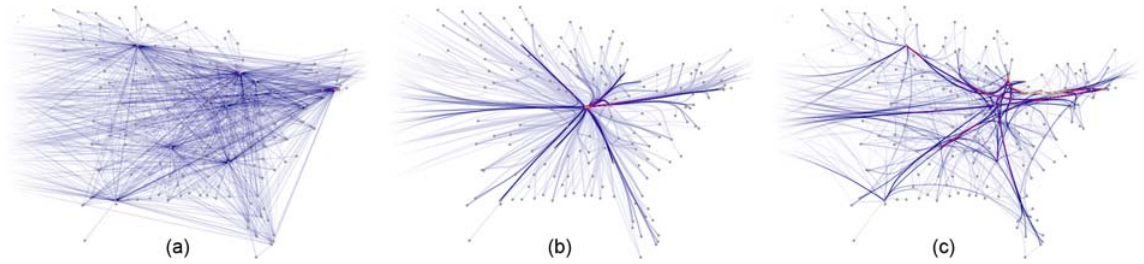

Figure 9 Bundling Visualization Adopted from Holten and Van Wijk (2009).

3 Research Objective

In this section, the main research question and sub-questions are presented, furthermore, the scope of the research (must, must not and could) is briefly introduced.

3.1 Research Question

Main Question: Is it possible to extend the space-filling curve clustering and indexing methods for trajectory data into a distributed database with MPP architecture?

1. How to perform trajectory modelling? Is it better to model it as a sequence instead of a point cloud or grid?
2. How to perform spatial accessing? Is it possible to use space-filling curve clustering and indexing methods for sequence-based modelling?
3. How to distribute data to the multiple nodes of the MPP database? Has the nature of SFC encoding changed, and how does it affect some regular queries?

The details of the questions (such as the parameters that need to be explored) are not shown in detail, but they can be seen in Section 4.

3.2 Research Scope

This research will be mainly divided into two parts: engineering implementation and scientific research. Among them, the engineering implementation as the basis of ev-

erything must be completed, including database deployment, Python-DBMS interface implementation, and basic applications such as visualization.

For each sub-problem, a minimum degree of scientific research must be completed, including the performance of one space-filling curve such as the Morton curve, the performance of one distribution strategy such as random distribution etc.

Detailed parameter tuning (such as comparisons of the Morton curve and Hilbert curve) in the production environment testing with extremely large volumes of data will be optional due to factors such as time and equipment. This research would not focus on the compression and the absolute speed (due to the different hardware and software environments to existing benchmarks).

Considering that the production environment is difficult to obtain, this study first considers testing on a virtual environment with a subset of data, focusing on the distribution of various query data to indirectly measure performance. The experiments on the production environment would be optional if time and other resources allowed.

4 Methodology

In this section, a brief methodology (shown in Figure 10) of this research is presented.

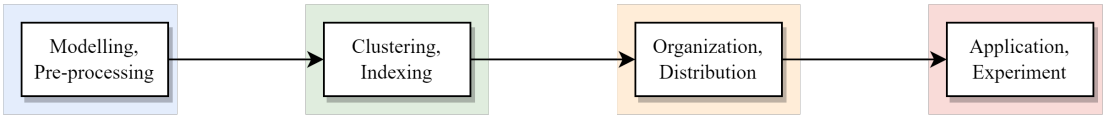

Figure 10 Methodology and Work Flow: the methodology of this research is divided into four main parts. Each part are one-to-one correspondence to previously mentioned related work and proposed research sub-questions.

Modelling/Pre-processing: This is in response to the question "How to perform trajectory modelling". This research will reconstruct the trajectory points into trajectory sequences, and then use the states to first split the whole trajectories irregularly, and then split them regularly using the space-time cube. The number of states used and the granularity with which the dimensions are divided will be explored as parameters affecting performance.

Clustering/Indexing: This is in response to the question "How to perform spatial accessing". This research will use space-filling curves (specifically the Morton curve) for clustering and encoding, and then use b-tree (or other linear indexing) methods for indexing. The scaling of different dimensions and the order in which they are encoded (XYT, XTY etc.) will be explored as parameters affecting performance (li2 (2020); Psomadaki (2016)).

Deployment/Distribution: This is in response to the question "How to distribute data to the MPP database". This research explores the possibility of random distribution,

distributed by space-filling code, distributed by vehicle ID, and distributed based on machine learning, the load balance is the final target (Gao et al. (2022)). Considering that virtual machines are used, the important point is that each selection needs to pull data from each node evenly.

Application/Experiment: How to measure the performance of modelling, indexing and distributing needs to be tested experimentally, and some common query applications will be implemented. At the same time, considering the characteristics of distributed systems, the integrity of the space-filling curve may be destroyed, and some queries may need to be changed accordingly (Liu (2022)).

5 Practical Issues

In this section, some practical matters related to this graduation research are presented, such as schedule, platforms, tools and data availability, which will support the feasibility of this research.

5.1 Time Planning

The proposed timeline (shown in Figure 11) is also presented, most of the tasks before P2 have been done (although some content may still need to be re-read and optimized in the future). Considering that there is still the possibility of adjustments to the research plan, this timeline only serves as a guide.

5.2 Platforms and Tools

1. Database Implementation: Greenplum, built on Postgresql, works as an open-source MPP database.
2. Experiment Platform: VMware Workstation, working together with Xshell, set up virtual machines on a single physical machine.
3. Pre-processing and Interaction: Python, along with its libraries such as Pandas eases the programming tasks.

5.3 Datasets Collection

Nine days' taxi data in Zhuhai (a city with 2 million population which is near the HongKong, China). There are 10GB of data with 20 million recodes and 3000 taxis per day. There are taxi_id, gps_time, latitude, longitude, altitude, speed, direction, mileage, state and other fields, a total of 66 attributes whose samples are shown in Table 2.

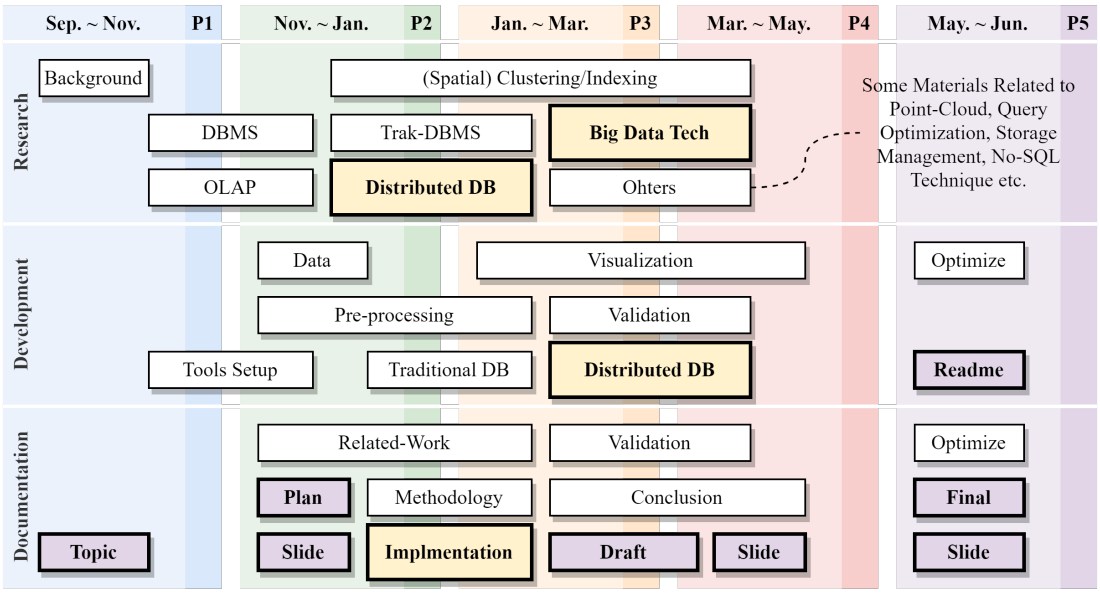

Figure 11 Gantt Chart for Time Planning: The timeline is divided into three parts: Research (reading relevant materials), Development (specific coding implementation) and Documentation (notes, thesis and slides). The key stages (with higher importance and would be allocated more time and resources) are marked in yellow. The deliverables at each stage are marked in purple.

Table 2 Datasets Sample

taxi_id	gps_time	latitude	longitude	state
800000000130	2021-10-14 00:00:03	22.22945	113.530618	2
800000000130	2021-10-14 00:00:13	22.228835	113.531048	2
800000000130	2021-10-14 00:00:23	22.228315	113.531411	2

In case of experiment need, the Automatic Identification System (AIS) [1] data is also prepared. It may share different properties with the taxi data, such as the spatial distribution (clustering), sample rates etc.

[1]The automatic identification system (AIS) is an automatic tracking system that uses transceivers on ships and is used by vessel traffic services (VTS).

References

Manage 4D historical AIS data by Space Filling Curve, author=Li, Jinglan. Master's thesis, Delft University of Technology, 2020.

Tariq Alsahfi, Mousa Almotairi, and Ramez Elmasri. A survey on trajectory data warehouse. *Spatial Information Research*, 2020.

Filip Biljecki, Hugo Ledoux, and Peter Van Oosterom. Transportation mode-based segmentation and classification of movement trajectories. *International Journal of Geographical Information Science*, 2013.

Irene De Vreede. Managing Historic Automatic Identification System Data by Using a Proper Database Management System Structure. Master's thesis, Delft University of Technology, 2016.

R Elmasri, SB Navathe, R Elmasri, and SB Navathe. *Fundamentals of Database Systems*. Springer, 2015.

Fan Gao, Peng Yue, Zhipeng Cao, Shuaifeng Zhao, Boyi Shangguan, Liangcun Jiang, Lei Hu, Zhe Fang, and Zheheng Liang. A multi-source spatio-temporal data cube for large-scale geospatial analysis. *International Journal of Geographical Information Science*, 2022.

Xuefeng Guan. *High-performance spatiotemporal computing and applications*. China Science Publishing (in Chinese), 2020.

Danny Holten and Jarke J Van Wijk. Force-directed edge bundling for graph visualization. In *Computer graphics forum*. Wiley Online Library, 2009.

Luca Leonardi, Gerasimos Marketos, Elias Frentzos, Nikos Giatrakos, Salvatore Orlando, Nikos Pelekis, Alessandra Raffaetà, Alessandro Roncato, Claudio Silvestri, and Yannis Theodoridis. T-Warehouse: Visual OLAP analysis on trajectory data. In *2010 IEEE 26th international conference on data engineering (ICDE 2010)*. IEEE, 2010.

Luca Leonardi, Salvatore Orlando, Alessandra Raffaetà, Alessandro Roncato, Claudio Silvestri, Gennady Andrienko, and Natalia Andrienko. A general framework for trajectory data warehousing and visual OLAP. *GeoInformatica*, 2014.

D Li. The intelligent processing and service of spatiotemporal big data. *Journal of Geo-Information Science*, 2019.

Wenwen Li, Michael Batty, and Michael F Goodchild. Real-time GIS for smart cities. *International Journal of Geographical Information Science*, 2020.

Haicheng Liu. *nD-PointCloud Data Management: continuous levels, adaptive histograms, and diverse query geometries*. PhD thesis, Delft University of Technology, 2022.

Ahmed R Mahmood, Sri Punni, and Walid G Aref. Spatio-temporal access methods: a survey (2010-2017). *GeoInformatica*, 2019.

Martijn Meijers and Peter van Oosterom. Clustering and Indexing Historic AIS Data With Space Filling Curves. Technical report, Delft University of Technology, 2018.

Martijn Meijers, Peter van Oosterom, and Wilko Quak. Management of AIS messages in a Geo-DBMS. Technical report, Delft University of Technology, 2016.

Nikos Pelekis, Elias Frentzos, Nikos Giatrakos, and Yannis Theodoridis. HERMES: A trajectory DB engine for mobility-centric applications. *International Journal of Knowledge-Based Organizations (IJKBO)*, 2015.

Dieter Pfoser, Christian S Jensen, Yannis Theodoridis, et al. Novel approaches to the indexing of moving object trajectories. In *VLDB*. Citeseer, 2000.

Styliani Psomadaki. Using a space-filling curve for the management of dynamic point cloud data in a relational DBMS. Master's thesis, Delft University of Technology, 2016.

Damião Ribeiro de Almeida, Cláudio de Souza Baptista, Fabio Gomes de Andrade, and Amilcar Soares. A survey on big data for trajectory analytics. *ISPRS International Journal of Geo-Information*, 2020.

Peter van Oosterom. Spatial access methods. *Geographical information systems*, 1999.

Peter van Oosterom and Tom Vijlbrief. The spatial location code. In *Proceedings of the 7th international symposium on spatial data handling, Delft, The Netherlands*, 1996.

Chunbo Wang. *Use Greenplum efficiently: getting started, advanced and middle platform*. China Machine Press (in Chinese), 2022.

Esteban Zimányi, Mahmoud Sakr, and Arthur Lesuisse. MobilityDB: A mobility database based on PostgreSQL and PostGIS. *ACM Transactions on Database Systems (TODS)*, 2020.