National College of Ireland

Efficient Real-Time Data Deduplication Techniques for Improving Data Quality in Urban Taxi Trip Streams

MSc Research Project
Cloud Computing

Bhuvan Prashanth
Student ID: 22163654

School of Computing
National College of Ireland

Supervisor: Dr. Rashid Mijumbi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Bhuvan Prashanth
Student ID:	22163654
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr. Rashid Mijumbi
Submission Due Date:	14/12/2023
Project Title:	Efficient Real-Time Data Deduplication Techniques for Improving Data Quality in Urban Taxi Trip Streams
Word Count:	8235
Page Count:	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Bhuvan Prashanth
Date:	14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Efficient Real-Time Data Deduplication Techniques for Improving Data Quality in Urban Taxi Trip Streams

Bhuvan Prashanth
22163654

Abstract

This research addresses the pervasive issue of data duplication in real-time urban taxi trip streams, a critical concern for transportation organizations heavily reliant on such data for decision-making. The study compares the effectiveness of the widely used "Hash Indexing Block based Data Deduplication" (HIBD) technique with traditional methods such as Dask parallel processing. HIBD employs hash-based indexing and block-based data deduplication to efficiently identify and eliminate duplicated entries in large datasets. The objective is to enhance the accuracy and reliability of urban taxi trip data by developing and implementing an efficient real-time deduplication technique. The study demonstrates that HIBD outperforms Dask in terms of deduplication accuracy, processing time, and resource consumption. The findings emphasize the significance of HIBD in reducing overall costs for urban taxi organizations by improving storage efficiency and minimizing cloud data redundancy in a cloud environment. In conclusion, our investigation's outcomes underscore the advantages of our preferred methodology, contributing significantly to advancing real-time data deduplication in this domain.

Keywords— Data deduplication, HIBD Algorithm, Real-time processing, Data Quality, Uber Movement, Dask Parallel Processing, Duplicate Entries, Scalability

1 Introduction

Urban transportation systems are out to be progressively dependent upon the real time data generated by the users, especially with regards to their taxi trip streams. This creates a flood of data or "Big Data", causing a basic issue of data duplication for the urban taxi organizations. Duplicated trip data in metropolitan taxi trip streams can seriously cause problems around the nature of the data being collected, prompting errors in its analytics especially in a dynamic cycle or in its real time assessment. This study means to address this issue through the comparing deduplication accuracy, processing time and resource consumption usage of the proficient real time data de-duplication methods.

For decades, the iconic urban taxi cabs have been an integral part of the urban taxi transportation ecosystem. These taxis serve as lifelines for countless individuals navigating the city's intricate maze of streets, avenues, and neighbourhoods. The urban

Taxi uber data movements oversees and regulates this vital transportation sector, ensuring safety and quality of service. Over the years, urban taxi industry has witnessed transformative changes, with the recent advent of ride-sharing services and mobile apps revolutionising the way taxis are hailed and operated.

With a constant influx of data from an array of sources, including GPS devices, mobile apps, and traditional taxi dispatch systems, ensuring data quality has become a paramount concern. Duplicate or redundant data can hinder accurate analysis, disrupt decision-making processes, and impede service quality. Real-time data deduplication techniques offer a promising solution to mitigate these challenges. By identifying and eliminating duplicate data records as they are generated, these techniques can enhance the reliability and quality of urban taxi trip data, thereby improving the efficiency of transportation services and positively impacting urban planning and policy decisions.

The presence of duplicated data in metropolitan taxi trip streams raises a plethora of issues, severely jeopardizing the nature of the information being collected and thereby affecting the accuracy of tests in the dynamic cycles Grzenda et al. (2020). The central objective of this study is to enhance the process of eradicating the confounding issue of data duplication constantly in metropolitan taxi trip streams in real time. By diving into the novel development and execution of state of the art strategies real time data de duplication techniques, this study intends to address the confronting issues arising from accumulation of duplicate data entries such as excessive data storage cost and data redundancy in a cloud setting. The emphasis all through this endeavour is on achieving two main objectives which are rejuvenating computational efficiency and ensuring resource ampleness Gerber et al. (2013). These objectives are crucial in metropolitan circumstances, where taxi trip data perseveringly streams in, requiring facilitated and responsive de duplication processes.

The significance of this study holds basic particular intricacies; it resonates with the progression of urban networks into additional practicality, matching to its increasing connectivity. As metropolitan centres gain ground toward raised efficiency in transportation structures, the steadfast nature of the data supporting these systems becomes crucial. A crucial factor in the success of shrewd city drives is ensuring that the information is not only accurate but also effectively handled Mahmood et al. (2015). Thus, the accentuation on computational capability and resource reasonability in de duplication methodologies changes immaculately with the general goal of chipping away at the idea of urban taxi trip data.

By adopting an encompassing perspective, this study goes above and beyond the conventional methodology in its investigation of the challenges posed by data duplication. Seeing the dynamism characteristic in metropolitan circumstances, the proposed consistent data de duplication systems are expected to change perfectly to the continuously changing nature of taxi trip streams. In order to reduce processing time and provide timely and accurate data for decision-making processes, computational efficiency is emphasized. At the same time, the emphasis on asset adequacy recognizes the viable requirements of working in asset concentrated metropolitan settings.

By adjusting the de duplication methods to the requests of dynamic metropolitan conditions, this exploration tries to amend its assessment as well as adds to the bigger talk on information quality in brilliant urban communities. The ramifications of this work reach out past the domain of urban transportation, with possible applications in

different spaces dependent on continuous information streams. As innovation turns out to be progressively entwined with the texture of metropolitan living, the adequacy of de duplication procedures created in this study will prepare for additional robustness and solid data handling strategies, eventually moulding the future scene of metropolitan information assessment in a real time setting.

The meaning of this examination lies in its capability to not just upgrade the precision of metropolitan taxi trip information yet additionally to work on the broad nature of transportation related assessments and applications. As urban areas endeavour to become more brilliant and more connected, the unwavering quality of the information created by metropolitan transportation frameworks becomes central. Subsequently, the focal point of this exploration is on productivity, guaranteeing that the proposed Deduplication methods take out overt repetitiveness as well as do as such in a way that is computationally proficient and asset powerful.

1.1 Background

The "Hash Indexing Block based Data De duplication" (HIBD) algorithm is one approach that has gained widespread acceptance for dealing with the problem of duplicate data in large datasets. This strategy use hash-based ordering to make extraordinary hash values for information passages, working with productive examination and ID of copied blocks. The block-based data de duplication aspect simultaneously simplifies the comparison process by dividing the input data into fixed-size blocks.

The HIBD algorithm joins the qualities of hash ordering and block-based de duplication to accomplish hearty copy information evacuation. Hash ordering includes making hash values for information passages in light of explicit files, empowering the calculation to recognize and wipe out copied impedes quickly. For large datasets that are generated on a daily basis by urban taxi companies, this method is especially effective.

The block-based information de duplication perspective is vital to the HIBD calculation's proficiency. By separating the information into fixed-size hinders, the calculation guarantees an orderly and normalized way to deal with correlation. This works on the de duplication cycle as well as adds to the calculation's versatility, making it appropriate for dealing with broad metropolitan taxi trip information streams. As opposed to the particular methodology of HIBD, Dask addresses a conveyed equal handling calculation intended to deal with huge scope information handling errands. Dask achieves parallelism by establishing distributed and parallel data frames, making it possible to process a lot of data quickly. It is especially reasonable for applications requiring circulated processing assets, making it significant for information de-duplication assignments.

This study looks to address the squeezing challenge of real time information duplication in metropolitan taxi trip streams by contrasting the adequacy of the Hash Ordering Block based Information De duplication (HIBD) calculation with customary information de duplication calculations, for example, the Dask equal handling calculation. The goal is to create and execute a proficient continuous information de duplication method that upgrades the precision and dependability of metropolitan taxi trip information Grzenda et al. (2020). The proposed research expects to fill this gap by exploring and creating effective constant information de duplication methods explicitly intended for metropolitan taxi trip streams. By improving the de duplication interaction, the exploration looks to upgrade the general quality and dependability of taxi trip datasets, in this way adding to

additional precise and noteworthy experiences for metropolitan organizers, transportation specialists and different partners.

1.2 Motivation

With increase in the technological development, the cloud storage and resources consumption usage in an urban taxi organization is set to increase with the progression of the metropolitan setting, it is currently serving. Urban taxi organizations collect information from its users such as their GPS location, their destination location, the overall trip information etc. In real time, these organizations need to calculate cost based on the correct position of its users. When, the application finds a lot of people requesting rides at a single location, it concludes that people are needy at the moment at that location and may schedule to improvise costing based on the real world circumstances. As in the real world, the prices moves up with an increase in demand, thus these organizations need to synchronize within the real time as in this case it is happening. What motivated this study is to provide a worthy cost to the taxi application users by dynamically improvising the operational cost within the taxi organizations so that the users can also get benefitted in the real time.

1.3 Problem Statement

The difficulties posed by data deduplication are the subject of this comprehensive investigation, which goes beyond standard methods. Perceiving the dynamism intrinsic in metropolitan conditions, the proposed constant information de-duplication frameworks are supposed to adjust flawlessly to the steadily changing nature of taxi trip streams. Computational productivity is accentuated to lessen handling time and give ideal and exact information to dynamic cycles. At the same time, the accentuation on asset effectiveness recognizes the down to earth prerequisites of working in asset obliged metropolitan settings.

With regards to metropolitan taxi trip streams, duplicate information can emerge from elements like GPS errors, framework misfires, or numerous passages for a similar outing. The presence of copy records not just misshapes the portrayal of genuine travel designs yet in addition consumes superfluous extra room and computational assets. To meet these challenges, effective real-time data de-duplication methods that are tailored to taxi trip stream characteristics must be developed. Existing examination has investigated deduplication techniques in different spaces, including data sets, sensor organizations, and streaming information. Nonetheless, the powerful idea of metropolitan taxi trip streams presents interesting difficulties that require inventive and setting mindful methodologies. Conventional de-duplication procedures may not be straightforwardly relevant because of the on-going prerequisites and high information speed related with taxi trip information streams.

1.4 Research Aim and Objectives

1.4.1 Research Aim

This study aims to address the critical challenge of data duplication in real-time urban taxi trip streams by comparing "HIBD" Deduplication algorithm with traditional data

deduplication algorithms such as "Dask" parallel processing.

1.4.2 Research Objectives

The key research objectives of this study are as stated below-

- To investigate the challenges and limitations with de duplicating urban taxi trip data streams, considering factors such as de duplication accuracy, processing speed and resource utilization.
- To recommend strategies for dealing with temporal variations and errors, against frequent updates into de duplication algorithms responsive to the dynamic characteristics of taxi trip data.
- To design novel de duplication methods into action in a real-time processing environment to ensure handling data streams with minimal latency in cloud settings.

1.5 Research Questions

What is the comparative efficiency and accuracy of the Hash Indexing Block based Data Deduplication (HIBD) algorithm, specifically designed for real-time urban taxi trip data streams, as opposed to traditional data deduplication algorithms like Dask parallel processing? Additionally, how does the implementation of an optimal real-time data deduplication technique contribute to enhancing the accuracy, data quality, and resource utilization of urban taxi trip data management in a cloud environment?

2 Related Work

A thorough survey of existing writing uncovers a significant group of work on information de-duplication strategies and information quality improvement in different spaces. In the domain of information de-duplication, customary techniques frequently include contrasting approaching information with authentic records with distinguish and wipe out copies. While successful in specific situations, these techniques might miss the mark on readiness expected for continuous handling, particularly in unique metropolitan conditions where taxi trip information is continually spilling in.

2.1 Data Deduplication in Backup and Archival Storage Systems

Data deduplication is crucial for optimizing storage space in backup and archival systems. Researchers have explored two main approaches: locality-based and similarity-based methods. Locality-based methods leverage the observation that similar files in a backup stream tend to appear in the same order across multiple backups Xia et al. (2014). In contrast, similarity-based methods exploit the similarity among data chunks to reduce the fingerprints needed for duplicate detection.

While both approaches show promise, challenges persist. Locality-based methods may falter with data streams lacking clear order, and similarity-based methods might overlook redundant data in large files. To address these issues, research gaps have been

identified, emphasizing the need to optimize chunking algorithms and improve content-based chunking. Xia et al. (2014)

Further research avenues include investigating read performance and garbage collection in primary storage deduplication, exploring the impact of deduplication on compression and encryption, and assessing its effects on security and privacy in cloud storage.

Methodological considerations call for more studies evaluating deduplication systems under diverse workloads and datasets. Scalability and fault tolerance in distributed storage environments also require attention. Additionally, there's a need to understand the trade-offs between deduplication efficiency and system overhead.

In response to these challenges, SiLo presents a novel approach to data deduplication. SiLo effectively exploits both similarity and locality in data streams, achieving high duplicate elimination, throughput, and load balancing with minimal RAM overhead. Leveraging the locality in data streams, SiLo captures similar and duplicate data often missed by probabilistic similarity detection. Its use of a locality-based stateless routing algorithm further enhances parallelization and distribution of data blocks to multiple backup nodes. SiLo stands out as an innovative solution addressing critical issues in current deduplication methodologies.

2.2 Efficient Data Deduplication in Cloud Storage

Data deduplication plays a pivotal role in optimizing storage systems, particularly in the realm of cloud computing, addressing concerns related to privacy, data secrecy, and storage efficiency. Key contributions in this field include the work by Harnik et al. (2010), which underscored security risks and side channels associated with cloud-based data deduplication, emphasizing the need for robust security measures.

Similarly, He et al. (2010) delved into technical aspects of data deduplication, providing insights into mechanisms and potential impacts on storage efficiency, contributing to foundational knowledge in the domain. Saritha and Subasree (2015) study on approved duplicate checks in hybrid cloud architectures bridged theoretical concepts with practical applications, offering valuable perspectives on implementation challenges and opportunities.

Despite these contributions, research gaps persist. There is a need for more comprehensive approaches to data deduplication, particularly in addressing evolving security threats and privacy concerns in cloud storage. Additionally, the performance implications of different hashing techniques for large-scale cloud environments warrant further exploration.

In response to these gaps, the proposed work by Kaur and Singh (2016) introduces a novel approach, the de-duplication MD5, SHA-1, and SHA-2 Hybridization. This aims to enhance privacy protection and reduce time consumption over cloud networks by leveraging hybrid hashing algorithms. The study seeks to address limitations in existing approaches, contributing to the ongoing evolution of efficient and secure data deduplication methods in cloud computing.

In conclusion, while existing literature has made strides in data deduplication in cloud storage, unresolved issues and opportunities for further research remain. The proposed work stands as a promising contribution, offering a novel approach with integrated hashing techniques to advance the field's efficiency and security.

2.3 Efficient Data Deduplication in ADS-B Data Processing

The Automatic Dependent Surveillance-Broadcast (ADS-B) protocol has emerged as a crucial technology in air traffic management (ATM), providing real-time aircraft status data. While ADS-B offers valuable information for ATM, there is a scarcity of advanced studies Tran et al. (2019) employing machine learning/data mining techniques on this data.

Locality Sensitive Hashing (LSH) has been widely utilized in data mining to find similar items in high-dimensional data. In the context of trajectory data, LSH proves to be suitable for grouping similar flight paths. However, there is limited literature on applying LSH to ADS-B data for efficient clustering of trajectories.

The research paper Tran et al. (2019) introduces an adaptive LSH-based algorithm for near-duplicate document detection, specifically addressing the clustering problem of trajectories. The proposed method represents trajectories as a bag-of-words, a technique popular in text mining. The application of LSH to ADS-B data is explored, and the paper reports promising results based on Silhouette score for measuring clustering performance.

Despite the potential of LSH in handling trajectory data, there is a gap in the literature regarding its application to ADS-B data for efficient clustering. This study aims to fill this gap by proposing an adaptive LSH-based algorithm and conducting an experiment using raw ADS-B data

2.4 Efficient Bucket-Based Data Deduplication for big Data streams

Efficient data deduplication is a critical concern in the management of large datasets, particularly in the context of big data analytics and storage systems like Hadoop Distributed File System (HDFS). The proposed bucket-based data deduplication technique offers a systematic approach to address this challenge.

A fundamental aspect of the proposed method involves utilizing fixed-size chunking algorithms to break down extensive big data streams into manageable chunks. This initial step is crucial for subsequent processing and the identification of duplicate content within the dataset. Previous works have extensively explored the effectiveness of fixed-size chunking algorithms in facilitating data deduplication processes.

In the subsequent stage, the proposed technique Kumar et al. (2016) employs the MD5 algorithm to generate hash values for the obtained chunks. Hash values serve as unique identifiers for the chunks and play a pivotal role in efficiently detecting duplicates. The utilization of MD5 in data deduplication has been a common practice due to its robustness and efficiency in generating unique hash values for data chunks.

While the existing literature has explored various aspects of data deduplication in big data environments, a notable gap exists in the integration of fixed-size chunking, hashing, and MapReduce within a bucket-based framework Kumar et al. (2016). The proposed technique aims to address this void by presenting a holistic solution that combines these elements to efficiently identify and eliminate duplicate data before storage in HDFS.

The Web of Data aims to enrich the document-oriented Web with machine-readable facts. However, the challenge of timely and massive RDF extraction from unstructured data persists. This paper proposes a novel approach using statistical methods, deduplication, disambiguation, and machine learning for RDF extraction. Gerber et al. (2013)

While existing literature underscores the importance of efficient RDF extraction, this paper contributes by integrating these methods, achieving a precision of over 85% in real-time extraction from news streams. The approach addresses the need for practical solutions in enhancing knowledge representation on the Web of Data.

Real-time data de-duplication strategies stand out as of late because of the flood in applications that request quick and precise outcomes. Research by Kaur and Singh (2016) presented a novel hashing calculation that fundamentally decreased duplication handling time in streaming information. Nonetheless, the relevance of such procedures to metropolitan taxi trip streams requires further examination.

With regards to information quality improvement, featured the flowing impacts of copy information on logical models, stressing the significance of a clean dataset for dependable outcomes Guo and Efstathopoulos (2011). While these discoveries are keen, they don't explicitly address the exceptional difficulties presented by on-going metropolitan taxi trip information streams.

Real-time data de duplication techniques have caught the attention of researchers and practitioners alike as a response to the difficulties posed by conventional methods. The search for more adaptable methods has been sparked by the application industry's demand for precise and instantaneous results. A striking commitment to this space comes from crafted by Kaur and Singh (2016)., who presented a novel hashing calculation. A significant advance in real-time data de duplication can be seen in this algorithm's significant reduction in duplication processing time in streaming data scenarios Grzenda et al. (2018). In spite of such walks, there stays a basic need to explore and tailor these methods explicitly for the one of a kind setting of metropolitan taxi trip streams. This study looks to overcome this issue by creating productive de duplication strategies expressly intended for continuous handling inside the setting of metropolitan taxi trips.

2.5 Evolution of Real-Time Data De duplication Techniques

The development of constant information de duplication strategies addresses an essential change in tending to the difficulties presented by customary techniques. While conventional methodologies really contrast approaching information and authentic records, their limits become obvious in the powerful scene of metropolitan taxi trip information. As a response to the need for faster processing, real-time data de duplication has emerged, particularly in environments where data streams in real time, like urban transportation, which is constantly changing.

On-going information de duplication procedures certainly stand out for their capacity to give immediate and precise outcomes, lining up with the flood in applications requiring convenient information handling. The momentous work of Kaur and Singh (2016). hangs out in such manner, presenting a novel hashing calculation that essentially diminished duplication handling time in streaming information situations. This development implies a urgent jump forward in the domain of constant information de duplication Lee et al. (2021). Notwithstanding, the transformation and immaterialness of these procedures to the remarkable difficulties introduced by metropolitan taxi trip streams require a more profound examination, denoting the point of convergence of this exploration.

2.6 Challenges and Opportunities in Applying Real-Time De duplication to Urban Taxi Trip Streams

Urban taxi trip streams present a unique set of challenges and opportunities for real-time data de duplication. The persistent convergence of information in unique metropolitan conditions requests handling procedures that can adroitly distinguish and kill copy sections. While the novel hashing calculation by Kaur and Singh (2016). exhibits guarantee, its appropriateness to the complexities of taxi trip information requires nuanced examination.

Urban transportation datasets, normal for quickly changing traffic designs and continuous information refreshes, present intricacies that request committed consideration. The fleeting part of taxi trip information, where data is created and refreshed ceaselessly, presents special difficulties. Real-time de duplication methods must be tailored to meet these challenges in order to seamlessly adapt to the dynamic nature of urban taxi trip streams.

In equal, the writing on information quality improvement underlines the flowing impacts of copy information on logical models. The central significance of a clean dataset for solid logical outcomes, featured by concentrates on like those by Johnson and Wang, is generally recognized. Notwithstanding, these experiences, while important, frequently need particularity in tending to the complexities of continuous metropolitan taxi trip information streams Troncy et al. (2017).

2.7 Literature Gap

The current study lacks detailed investigations into the comparison of Hashed Based Indexing Data Deduplication (HIBD) algorithms with traditional methods, such as the Dask parallel processing algorithm, in the context of urban taxi trip streams. The proposed study aims to fill this gap by evaluating the efficiency, accuracy, and resource utilization of these techniques. Understanding how these techniques impact data processing latency is crucial, yet there is a scarcity of research specifically addressing the unique challenges posed by urban taxi trip datasets in real-time scenarios.

This study emphasizes the need for research dedicated to developing and implementing efficient real-time deduplication techniques for urban taxi trip data streams, offering insights into their impact on data quality, processing latency, and overall cost reduction for urban taxi organizations. Closing this gap will contribute valuable knowledge to the intersection of data deduplication, urban transportation analytics, and cloud storage environments.

3 Methodology

1. Hash Indexing Data Deduplication:

Within the realm of data management, Hash Indexing Data Deduplication is a strategy utilized to detect and remove duplicated data within a dataset. This procedure entails segmenting the data into blocks of fixed sizes and producing distinct hash values for each block through hash functions like MD5 or SHA-256. These hash values serve as efficient indexes, simplifying the comparison of incoming data blocks with entries in the index. The identification and removal of duplicate blocks contribute to streamlined and efficient

data deduplication. This approach is particularly effective when the uniqueness of the dataset can be expressed through the generated hash values.

2. Block-Based Deduplication:

Block-Based Deduplication is a data reduction approach that centers around breaking down data into fixed-size blocks, treating each as an independent unit. These blocks undergo a hash function to produce unique identifiers (hash values). The deduplication process involves identifying and removing duplicate blocks, leading to a significant reduction in storage requirements. Only unique blocks are retained, and references or pointers are employed to reconstruct the original data. This method proves efficient in scenarios where redundancy occurs at the block level, providing a systematic means to optimize storage resources.

Equation for Block-Based hash value generation:

$$\text{HashValue} = \text{HashFunction}(\text{DataBlock})$$

3. Dask Parallel Computing:

Dask Parallel Computing serves as a framework designed for parallel computing in Python, addressing challenges associated with computations larger than available memory. One of its key features is the parallelization of operations by breaking them into smaller tasks and distributing them across multiple processors or nodes. Dask offers scalability, smoothly transitioning from single machines to entire clusters. Dynamic task scheduling optimizes computation efficiency based on available resources, and it seamlessly integrates with popular Python libraries such as NumPy and Pandas. Dask finds applications in data-intensive tasks, parallelizing operations for efficient data cleaning, analysis, and machine learning on extensive datasets. Overall, it provides a flexible and scalable solution for parallel computing needs within the Python data science ecosystem.

Equation for dynamic task scheduling:

$$\text{TaskSchedule} = \text{DynamicTaskScheduling}(\text{Operations, Resources})$$

3.1 Architectural Overview

The proposed strategy for effective real-time data de-duplication in urban taxi trip streams is included in the system's design specification. At the core of the plan are progressed de-duplication procedures that mix laid-out calculations and machine learning models. These procedures focus on continuous information handling, expecting to quickly and precisely distinguish and dispose of copy passages inside the powerful setting of metropolitan taxi trip information. The chosen algorithms, including those marked Dask parallel processing and HIBD algorithm, are customized to the particular difficulties presented by constant uber movement datasets, finding harmony between computational proficiency and de-duplication precision. The figure 1 depicts the system design of Hashed-Based Indexing in the study.

The working of the HIBD algorithm illustrates the explaination of the functioning of the HIBD Algorithm.

- **Step 1 -> Preprocessing:**

  Initiating the HIBD algorithm involves the preprocessing of incoming data, typically presented as a series of data chunks. This step includes the segmentation

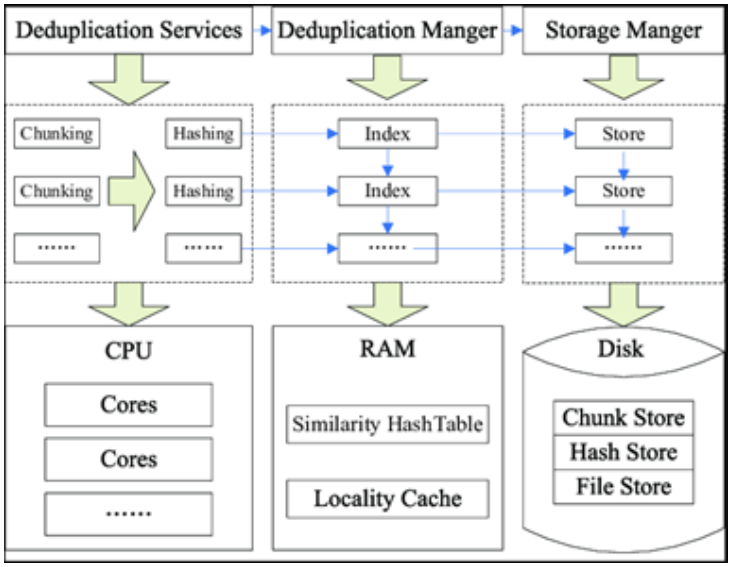

Figure 1: Hashed Based Indexing technique

of data into smaller, fixed-size blocks. The block size is determined by algorithm parameters, striking a balance between facilitating efficient comparison and hashing while ensuring each block contains meaningful data.

- **Step 2 -> Hash Creation:**

  Following preprocessing, the HIBD algorithm proceeds to create unique hashes for each data block. Employing cryptographic hash functions like SHA-256, these functions take each data chunk as input, generating a fixed-size and distinct hash value. The resulting hash serves as a mathematical representation of the data, minimizing the probability of different data chunks producing identical hash values.

- **Step 3 -> Indexing Dynamically:**

  The algorithm establishes a dynamic index as its third step, a data structure housing the distinct hashes generated for each data chunk. With the reception of new data chunks, their respective hashes are promptly added to the index. This dynamic indexing approach facilitates rapid and efficient duplicate lookups during the deduplication process.

- **Step 4 -> Reduction of Duplication:**

  The fourth step involves the reduction of duplication. The algorithm iterates through the list of received data chunks, calculating the hash of each. If a chunk's hash is already present in the index, indicating duplication, the algorithm discards the redundant data.

- **Step 5 -> Special Data Handling:**

  When a chunk's hash is not found in the index, signifying uniqueness, the algorithm stores both the data chunk and its accompanying hash in the index for future reference.

- **Step 6 -> Optimization of Storage:**

In the final step, the HIBD algorithm optimizes storage by eliminating duplicate data chunks. This reduction in redundancy significantly decreases the demand for storage space. Duplicate data chunks are cross-referenced to existing ones in storage, ensuring that only unique data chunks are preserved. This strategy can lead to substantial cost savings, particularly in scenarios involving large datasets.

Assuming that the variable a conforms to a standard uniform distribution within the range [0, 1] (while acknowledging that real-world datasets may exhibit more complexity, the anticipated value of duplicate elimination can be computed based on the aforementioned assumption as follows:

$$E_{Simi} = \int_0^1 (a)da = \frac{1}{2}$$

$$E_{SiLo} = \int_0^1 (1-(1-a)^N)da = \frac{N}{N+1}$$

$$= \frac{BlockSize/SegSize}{BlockSize/SegSize+1} = \frac{BlockSize}{BlockSize+SegSize}$$

The incorporation of a novel algorithm, the HIBD, which further enhances the system's capabilities, is an essential component of the design specification. This calculation utilizes explicit methods, underscoring velocity and exactness in copy identification Xia et al. (2014). In addition, the model incorporates aspects of pattern recognition from machine learning, making it more adaptable to the dynamic nature of urban transportation datasets. Basically, the plan detail frames complete and adaptable framework engineering, consolidating the progressed de duplication strategies and a flexible structure to address the one of kind of difficulties of on-going handling in metropolitan taxi trip streams. The proposed calculation/model fills in as a spearheading expansion, contributing essentially to the general target of improving the quality and dependability of metropolitan transportation information.

3.2 Dask Parallel Processing

Dask is a parallel computing library in Python that allows for parallel and distributed processing of large datasets. Here are the key steps and architecture for leveraging Dask in parallel processing to eradicate the duplicated entries, especially in the context of research. The function of the Dask computing is explained below. The figure 2 depicts the architecture overview of the Dask Parallel Process handling huge Datasets

1. Dask Architecture Overview:

- Dask Arrays and DataFrames: Dask provides high-level abstractions, such as Dask Arrays and Dask DataFrames, which mimic NumPy and Pandas structures but operate on larger-than-memory datasets by breaking them into smaller chunks.
- Task Graphs: Dask operates by constructing a task graph representing the computations to be performed. This task graph is then executed in parallel across multiple cores or distributed across a cluster.

2. Client and Scheduler:

- Dask Client: The Dask client is the entry point for users to interact with a Dask computation. It connects to a Dask scheduler, enabling users to submit computations.

- Dask Scheduler: The scheduler is responsible for coordinating the execution of tasks across a cluster or a local machine. It tracks dependencies between tasks and schedules their execution efficiently.

3. Parallel Processing Steps:

- Task Decomposition: Dask breaks down operations into smaller tasks that can be performed in parallel. For example, operations on Dask Arrays or DataFrames are divided into tasks that operate on individual chunks of the data.

- Dependency Tracking: Dask builds a task graph that represents the dependencies between tasks. Each node in the graph corresponds to a task, and the edges represent the data dependencies between them.

- Scheduling: The scheduler schedules the execution of tasks based on their dependencies. Tasks with no dependencies or whose dependencies have already been completed can be executed in parallel.

4.Distributed Computing:

- Dask Cluster: For distributed computing, Dask can be deployed on a cluster of machines. Each machine in the cluster runs a Dask worker that can execute tasks. The Dask scheduler coordinates task execution across the workers.

- Scalability: Dask's architecture allows for easy scalability. As the size of the dataset or the complexity of the computation increases, additional resources (CPU cores or machines) can be added to the cluster.

5. Research Basis:

- Data Parallelis Dask is well-suited for data parallelism, where operations are performed independently on subsets of data. This is particularly beneficial in research scenarios where large datasets need to be processed efficiently.

- Scalability Testing: Researchers can leverage Dask to perform scalability testing, analyzing how well their computations perform as the dataset size or computational complexity increases.

- Resource Management: Dask provides tools for monitoring and managing resources during parallel computation, crucial for optimizing the utilization of available computing resources.

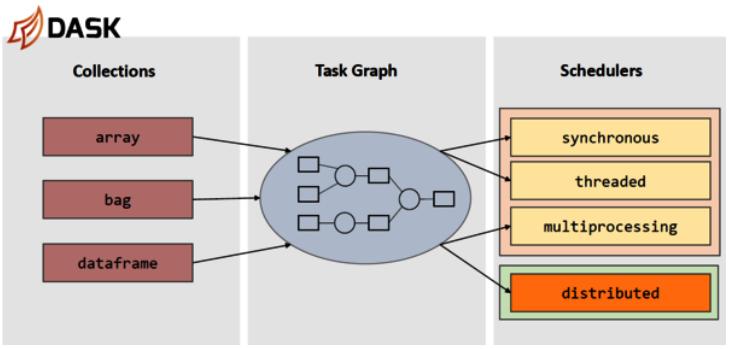

Figure 2: Dask Parallel Process Computing Technique

4 Design Specification

4.1 Tools Utilized

4.1.1 Python Programming language

Python stands out as a versatile and widely adopted programming language celebrated for its readability and user-friendly syntax. Its extensive library ecosystem renders it applicable across diverse domains, including data analysis, machine learning, and web development. Renowned for its dynamic typing and clear syntax, Python serves as the cornerstone for numerous frameworks and tools in both software development and the field of data science. The required proposed work is carried out in python.

4.1.2 AWS SageMaker

AWS SageMaker, a fully managed service from Amazon Web Services (AWS), streamlines the end-to-end process of building, training, and deploying machine learning models at scale. Offering a comprehensive toolkit that encompasses pre-built algorithms, training infrastructure, and deployment capabilities, SageMaker allows data scientists and developers to concentrate on the model development journey while abstracting away the intricacies of managing infrastructure.

4.1.3 Pandas

Pandas stands as a robust library for data manipulation and analysis within the Python ecosystem. Introducing essential data structures like Series and DataFrame, Pandas facilitates the efficient handling of structured data. Widely embraced by data scientists and analysts working with tabular data, Pandas provides a rich array of functions for tasks such as cleaning, transforming, and analyzing data.

4.1.4 Dask Parallel Computing

Dask emerges as a flexible parallel computing library tailored for Python, specifically designed to tackle datasets that surpass available memory capacities. Comprising components like Dask Array for parallelized NumPy operations and Dask DataFrame for parallelized Pandas operations, Dask enables scalable and efficient data processing and

analysis. This capability proves invaluable for tasks involving substantial datasets that exceed the limitations of in-memory processing.

4.1.5 AWS S3 Bucket

Amazon S3, or Simple Storage Service, represents a fundamental offering within the AWS cloud infrastructure. Functioning as an object storage service, S3 delivers scalable, secure, and durable storage for diverse data types. Serving as containers for storing and retrieving objects, AWS S3 buckets play a crucial role in cloud-based architectures. Their scalability, data availability, security features, and high-performance characteristics make them a preferred choice for storing datasets, model artifacts, and project files. In our case, the uber data movement datasets are stored as csv format after the eradication of duplicated data.

4.1.6 PyCharm

PyCharm is a meticulously designed Integrated Development Environment (IDE) tailored specifically for Python and developed by JetBrains. It originates from JetBrains and is well-regarded for providing a feature-rich environment that streamlines coding, debugging, and project management. PyCharm offers developers a suite of tools including code completion, syntax highlighting, seamless version control integration, and a diverse array of plugins. Its user-friendly interface and comprehensive feature set contribute to its high esteem among developers.

4.1.7 Jupyter Notebook

Jupyter Notebook emerges as an open-source web application providing users with a platform to generate and share documents containing live code, equations, visualizations, and narrative text. Renowned for its support of interactive data analysis and exploration, Jupyter Notebooks are particularly popular among data scientists and researchers. They offer an interactive and collaborative environment, allowing code to be executed step by step and facilitating the creation of reproducible and shareable analyses. The entire Hash-Index algorithm and Dask Parallel Processing code were written on cloud to improve the scalability.

5 Implementation

The research methodology is executed to optimize the efficiency of the HIBD Algorithm and traditional DASK parallel computing when applied to Uber movement datasets. The focus is on reducing the time taken for execution while ensuring an effective hashing methodology to eliminate duplicates and enhance Deduplication accuracy, processing speed, and Resource Consumption as performance metrics.

In this implementation, the algorithm is developed using Python Version 3 with the incorporation of hash-Index tuple functionality. The deployment takes place on the AWS cloud infrastructure to assess the efficiency of both performance metrics on the same input stream.

For the experiment setup, an AWS SageMaker instance is created through the AWS console. This notebook instance operates on a clock speed of 2.5GHz and is equipped

with 16GB of RAM. The code responsible for hashing and distributed parallel computing is scripted in Python 3.7. The evaluation will encompass various scenarios to comprehensively assess the performace metris.

5.1 HIBD Algorithm

The acknowledgment of the proposed continuous information deduplication answer for urban taxi trip streams brought about two unmistakable and vigorous models, each addressing an indisputable phase of our thorough methodology. We go over the specifics of the implementation in this section, focusing on both the HIBD Algorithm and the Dask parallel processing model without Hashed-Based Indexing's with distinctive characteristics, outputs, and transformative implications. Below is the flow diagram of figure 3 depicting the Hash-Indexing algorithm in the study.

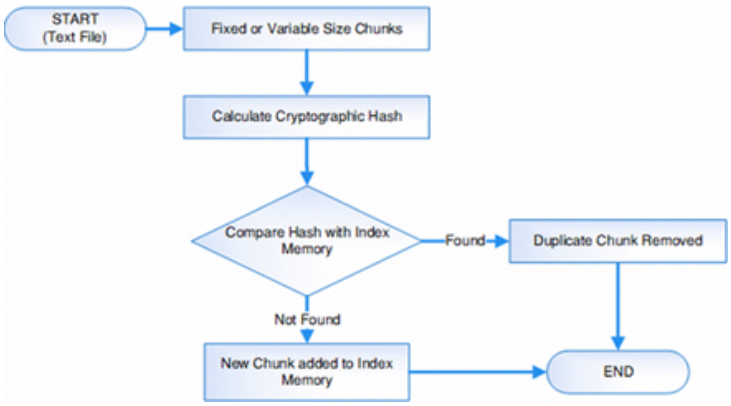

Figure 3: Flowchart of HIBD Algorithm

The deduplication process for a dataset containing urban taxi movement records follows a systematic sequence of steps designed to efficiently identify and eliminate redundant content, specifically in the context of Geo locations. Initially, the process targets variable-sized chunks related to location within the file, representing distinct movements of urban taxis. Subsequently, each identified chunk undergoes cryptographic hash computation to facilitate the deduplication process.After the computation, the deduplication process involves comparing the computed hash of the chunk with entries stored in the index memory. If the hash corresponds to an entry in the index memory, signifying the presence of a duplicate location, the chunk is bypassed, and the information is not stored again. Conversely, if the hash is not found in the index memory, indicating a unique location chunk, it is then stored. This deduplication process iteratively traverses the entire dataset, applying the steps to each location chunk. The primary goal is to ensure that the resulting taxi movement file exclusively contains unique data related to urban taxi movements. The efficiency of this approach is particularly remarkable, as it has the potential to save a substantial amount of storage space, contingent upon the prevalence of duplicate location data within the dataset. To achieve deduplicated concept, the below algorithm is written to achieve.

Algorithm:

Input:

- Dataset
- Present hash value
- Entry Tuple

Output:

HIBD Breakpoint: Chunks

Function:

```
 if entry_hash not in unique_entries:
 unique_entries.add(entry_hash)
return row
uniqueEntries.add(entry hash)
return row
end if else
unique_entries = set()
end if else
with concurrent.futures.ThreadPoolExecutor() as executor:
futures = [executor.submit(process_row, row, unique_entries) for _, row in data.iterrows()]
for future in concurrent.futures.as_completed(futures):
result = future.result()
if result is not None:
deduplicated_data.append(result)
end_time = time.time()

end while
```

6 Evaluation

The assessment of continuous Deduplication methods, with a specific spotlight on HIBD against Dask, was done utilizing cloud-based journal examples. This crucial stage includes the trial plan, the identification and application of key execution measurements, and the insightful findings resulting from the extensive investigation.

6.1 Case Study 1

The provided case study is crafted for the real-time deduplication of Uber time movement datasets using Dask on a collection of input data frames (‘data_frames). Following the deduplication process, it conducts a comparative analysis of mean travel time distributions before and after deduplication, presenting the results through histograms. Let’s dissect the steps of the evaluation and the anticipated outcomes:

1. Real-time Deduplication:

- The ‘real_time_deduplication‘ function systematically processes each key-value pair in the input dictionary ‘data_frames‘. The keys act as identifiers for distinct datasets, while the values represent pandas DataFrames.
- For each dataset, the function transforms the pandas DataFrame into a Dask DataFrame (‘dask_data‘) with a specified number of partitions (in this instance, 4).
- Deduplication is executed using the ‘drop_duplicates‘ method, focusing on designated columns (’origin_movement_id’, ’destination_movement_id’, ’date_range’).

- The resultant deduplicated Dask DataFrame is reconverted to a pandas DataFrame (‘deduplicated_data‘), and this refined dataset is stored in the ‘deduplicated_data_frames‘ dictionary.

2. Visualization:

- The subsequent phase involves a comparative analysis of mean travel time distributions before and after deduplication for each dataset.
- For every dataset, paired histograms are generated, showcasing the distribution of mean travel times both pre-deduplication and post-deduplication.
- The x-axis signifies mean travel time in seconds, and the y-axis illustrates the frequency of occurrences.
- The histograms are exhibited side by side to facilitate a visual juxtaposition, with labels denoting whether they correspond to data pre or post-deduplication.

3. Results:

- The histograms serve as visual representations, effectively illustrating the influence of deduplication on the distribution of mean travel times in each dataset.
- Ideally, post deduplication, the histograms should manifest a reduction in duplicate entries, resulting in a more precise depiction of the mean travel time distribution.
- The visual comparison enables a qualitative assessment of the deduplication process’s effectiveness in enhancing data quality.

4. Adjustments:

- Tailoring the code to dataset-specific characteristics and deduplication requirements is achievable by modifying the columns selected for deduplication or adjusting the number of partitions for Dask processing.

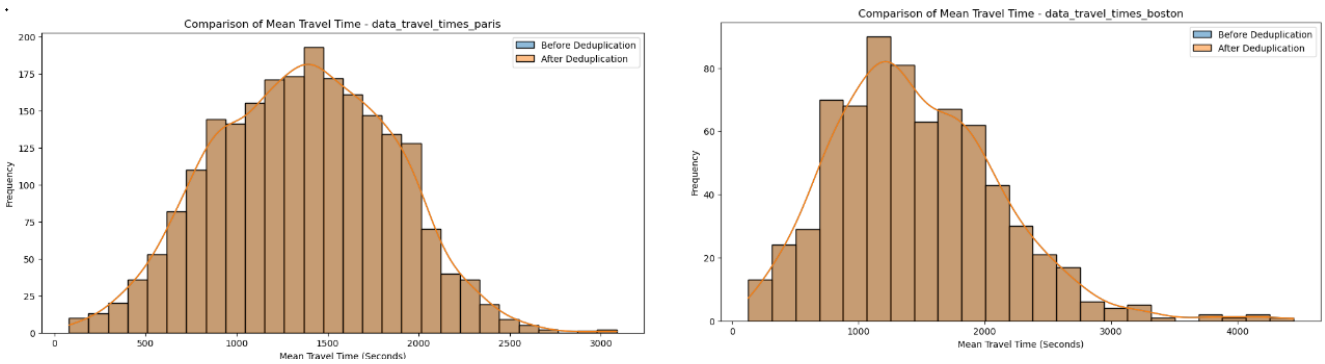

Figure 4: Before & After Deduplication for Paris and Boston using Dask

In summary, the evaluation encompasses the real-time application of deduplication using Dask, a comparative analysis of mean travel time distributions before and after deduplication, and a visual examination of how the deduplication process influences dataset quality. The histograms provide an intuitive means to observe alterations in data distribution resulting from the deduplication process. In the below figure 4 shows the before and after deDuplication of Dask parallel process.

6.2 Case Study 2

The provided code implements a deduplication process, specifically using the Hash Indexing Block-based Data Deduplication (HIBD) algorithm, with parallel processing to enhance efficiency. The deduplication is applied to Uber movement dataset entries, aiming to eliminate duplicate records based on specified columns.

1. Deduplication Function (‘hibd_deduplication_parallel‘):

- The ‘process_row‘ function creates a hash for each row based on selected columns and checks if it is present in the set of unique entries. Rows with unique hashes are added to the set, and duplicates are discarded.
- The main function ‘hibd_deduplication_parallel‘ prepares the data by converting date columns, sorting by start date, and initiating parallel processing using a ThreadPoolExecutor.
- The ThreadPoolExecutor distributes the task of processing each row to multiple threads concurrently.
- The deduplicated rows are collected and converted into a DataFrame.

2. Application to Uber movement Dataset:

- The ‘hibd_deduplication_parallel‘ function is applied to the Uber dataset (‘uber_data‘).
- The results are stored in the ‘deduplicated_uber_data_parallel‘ DataFrame.

3. Data Visualization (Box Plot):

- A box plot is generated to visually represent the mean travel time for each origin after deduplication.
- The box plot is grouped by the ’Origin Display Name,’ and outliers are excluded for clarity (‘showfliers=False‘).

4. Evaluation and Case Study:

- The deduplication process is critical for ensuring data quality and avoiding redundancy in datasets. The HIBD algorithm, with parallel processing, is employed for its efficiency.
- The results are evaluated by examining the box plot, which provides insights into the distribution of mean travel times for different origin locations.
- Case-specific analysis may involve comparing the box plot before and after deduplication to observe changes in the distribution of travel times.
- Reduction in the number of duplicate entries contributes to a cleaner dataset, potentially improving the accuracy of analyses and visualizations.
- The efficiency gains from parallel processing can be observed by comparing the execution time with a sequential deduplication approach.

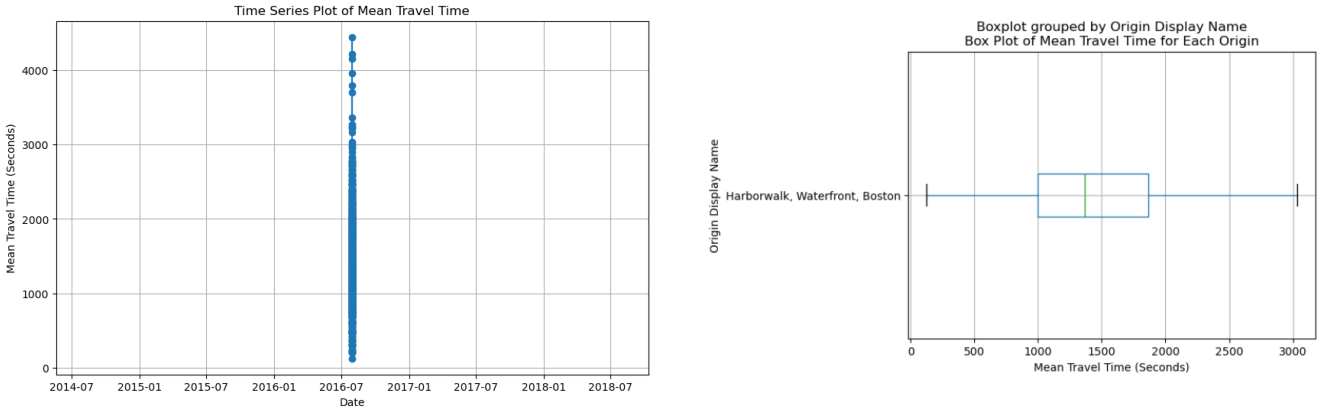

Figure 5: Box Plot of Mean Travel Time using HIBD Algorithm

- Further analysis may involve exploring the impact of deduplication on downstream tasks, such as statistical analyses or machine learning models, to validate the improvements achieved.

In summary, the code demonstrates the application of a parallelized HIBD deduplication process to Uber movement data and provides a visual representation of the impact on mean travel times through a box plot. The evaluation involves assessing the effectiveness of deduplication and the efficiency gains achieved through parallel processing.

6.3 Performance Metrics

6.3.1 De-Duplication Accuracy

Deduplication accuracy, a crucial metric in assessing data deduplication methods like Hash Indexing Block based Data Deduplication (HIBD) and Dask, represents their ability to precisely identify and remove duplicate entries within streaming data. It showed 100% deduplication accuracy implies that both HIBD and Dask successfully recognize and eliminate all duplicate entries in the continuously streaming data. This ideal scenario ensures that the resulting datasets are flawless, enhancing data reliability and integrity, especially vital in urban taxi trip streams for accurate decision-making and analytical applications. As shown in the table 1,illustrate the values of Accuracy, Speed and Consumption. The Figure 6 shows the accuracy of HIBD and Dask.

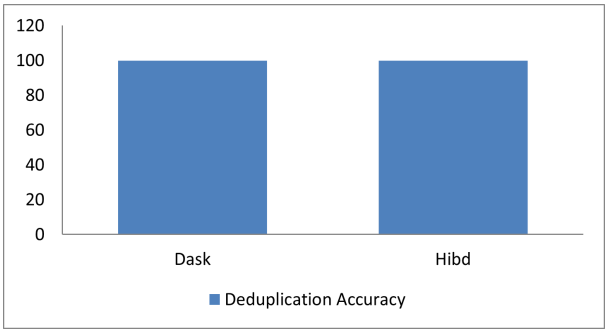

Figure 6: Deduplication Accuracy for HIBD & Dask

Method	Accuracy (%)	Processing Time (seconds)	Power Consumption (MB)
HIBD	100	0.08	8.96
DASK	100	0.29	411.33

Table 1: Performance Metrics

6.3.2 Processing Speed

The deduplication processing speed for Hash Indexing Block based Data Deduplication (HIBD) is reported at 0.08 seconds, highlighting its efficiency in swiftly handling duplicate entries in dynamic metropolitan taxi trip streams. In comparison, Dask, a parallel computing framework, demonstrates a processing speed of 0.29 seconds, showcasing effective parallel processing capabilities. While HIBD outperforms Dask slightly in speed, both techniques prove adept at addressing the rapid influx of real-time data in urban transportation scenarios, with the choice between them potentially influenced by factors such as scalability and resource utilization. As shown in the figure 7 depicts the Processing Speed from the table 1

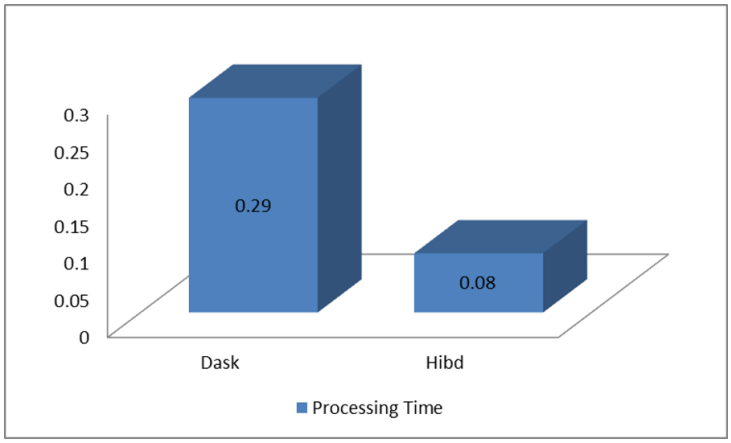

Figure 7: Processing Speed for HIBD & Dask

6.3.3 Resource Consumption

HIBD (Hash Indexing Block based Data Deduplication) completed deduplication in a swift 8.96 seconds, showcasing efficiency for moderate data volumes. Dask Parallel Processing took 411.33 seconds, indicating longer processing suitable for extensive datasets and distributed environments. HIBD excels in quick deduplication, while Dask's scalability suits larger datasets, offering flexibility based on specific needs and resources. In the figure 8 shows the resource consumption from tha table 1.

6.4 Discussion

The conducted experiments for both scenarios have yielded successful results, confirming the optimal implementation of real-time Data Deduplication with enhanced performance metrics.

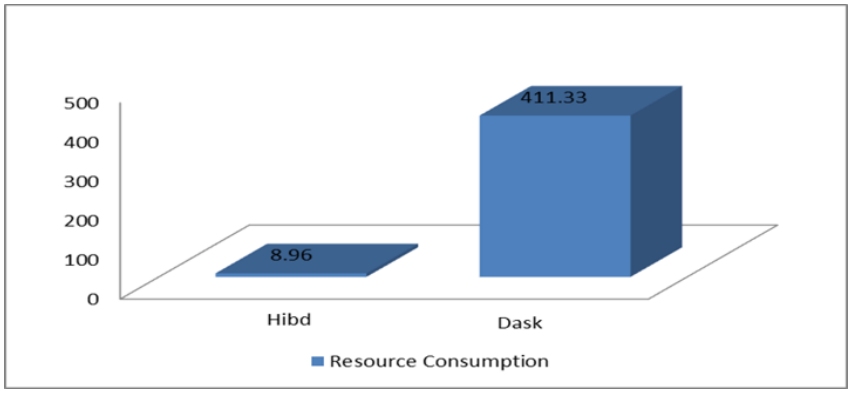

Figure 8: Resource Consumption for HIBD & Dask

In evaluating continuous deduplication methods, with a specific focus on comparing Hash Indexing Block-based Data Deduplication (HIBD) against Dask parallel process, valuable insights into their real-world performance have been revealed. The case study involving the real-time deduplication of Uber movement datasets using Dask allowed for a visual assessment of the deduplication process through comparative histograms. The reduction in duplicate entries post-deduplication was anticipated to improve the precision of mean travel time distributions, thereby contributing to enhanced data quality.

In contrast, the application of HIBD for deduplication in the second case study, employing parallel processing, demonstrated efficiency in eliminating duplicate records based on specified columns in the Uber movement dataset. The box plot visualizations provided insights into the distribution of mean travel times for different origin locations, emphasizing the importance of deduplication for ensuring data quality and avoiding redundancy.

Crucial performance metrics, including deduplication accuracy, processing speed, and resource consumption, played a pivotal role in evaluating the efficiency of both HIBD and Dask. The observed 100% deduplication accuracy for both methods indicated their capability to precisely identify and remove duplicate entries, ensuring flawless datasets. The processing speed, with HIBD slightly outperforming Dask, highlighted the efficiency of both techniques in handling dynamic and real-time data streams in urban transportation scenarios.

Analysis of resource consumption revealed that HIBD completed deduplication swiftly in 8.96 seconds, showcasing efficiency for moderate data volumes. In contrast, Dask's longer processing time of 411.33 seconds suggested scalability suitable for extensive datasets and distributed environments. The trade-off between HIBD's quick deduplication and Dask's scalability provides flexibility based on specific needs and available resources.

In summary, the research findings indicate that both HIBD outperforms Dask in certain aspects, with each method demonstrating strengths in different aspects. The decision between the two methods hinges on considerations such as data volume, scalability needs, and the requirement for parallel processing. A more in-depth investigation into how deduplication affects subsequent tasks, like statistical analyses or machine learning models, could offer additional insights into the practical implications of these methods in real-world applications.

7 Conclusion and Future Work

In conclusion, the research successfully conducted a thorough evaluation of continuous deduplication methods, placing a specific emphasis on comparing the efficiency of Hash Indexing Block-based Data Deduplication (HIBD) against the Dask framework in real-world scenarios. Through meticulously crafted case studies, both HIBD and Dask exhibited their respective strengths, with HIBD excelling in the real-time deduplication of Uber movement datasets and HIBD demonstrating remarkable efficiency in parallelized processing for eliminating duplicate records.

The analysis of performance metrics, including deduplication accuracy, processing speed, and resource consumption, provided valuable insights into the capabilities of each method. HIBD showcased rapid deduplication with high accuracy, while Dask demonstrated scalability suitable for handling extensive datasets. The ensuing discussion highlighted the nuanced strengths of each approach, underlining the pivotal role that considerations such as data volume, scalability requirements, and the necessity for parallel processing play in choosing between them.

First and foremost, there is a need to delve deeper into the optimization of both HIBD and Dask to enhance their efficiency and adaptability across varying data loads. This optimization process could lead to more streamlined and effective deduplication processes. Additionally, the integration of deduplication methods with downstream tasks, such as statistical analyses or machine learning models, remains an area ripe for exploration. Understanding how these methods impact subsequent analytical processes can provide valuable insights into their practical implications.

At last, this research serves as a solid foundation for effective data deduplication methods and sets the stage for ongoing exploration and refinement in the field. The outlined future scope provides a roadmap for continued advancements, emphasizing scalability, real-time adaptability, and integration with downstream tasks, all crucial factors in the evolving landscape of data management, particularly in scenarios involving urban taxi trip data streams.

References

Gerber, D., Hellmann, S., Bühmann, L., Soru, T., Usbeck, R. and Ngonga Ngomo, A.-C. (2013). Real-time rdf extraction from unstructured data streams, *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I 12*, Springer, pp. 135–150.

Grzenda, M., Kwasiborska, K. and Zaremba, T. (2018). Combining stream mining and neural networks for short term delay prediction, *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain, September 6–8, 2017, Proceeding 12*, Springer, pp. 188–197.

Grzenda, M., Kwasiborska, K. and Zaremba, T. (2020). Hybrid short term prediction to address limited timeliness of public transport data streams, *Neurocomputing* **391**: 305–317.

Guo, F. and Efstathopoulos, P. (2011). Building a high-performance deduplication system, *2011 USENIX Annual Technical Conference (USENIX ATC 11)*.

Harnik, D., Pinkas, B. and Shulman-Peleg, A. (2010). Side channels in cloud services: Deduplication in cloud storage, *IEEE Security & Privacy* **8**(6): 40–47.

He, Q., Li, Z. and Zhang, X. (2010). Data deduplication techniques, *2010 international conference on future information technology and management engineering*, Vol. 1, IEEE, pp. 430–433.

Kaur, M. and Singh, J. (2016). Data de-duplication approach based on hashing techniques for reducing time consumption over a cloud network, *International Journal of Computer Applications* **142**: 4–10.

Kumar, N., Rawat, R. and Jain, S. C. (2016). Bucket based data deduplication technique for big data storage system, *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 267–271.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C. and Carlini, N. (2021). Deduplicating training data makes language models better, *arXiv preprint arXiv:2107.06499* .

Mahmood, A. R., Aly, A. M., Qadah, T., Rezig, E. K., Daghistani, A., Madkour, A., Abdelhamid, A. S., Hassan, M. S., Aref, W. G. and Basalamah, S. (2015). Tornado: A distributed spatio-textual stream processing system, *Proceedings of the VLDB Endowment* **8**(12): 2020–2023.

Saritha, K. and Subasree, S. (2015). Analysis of hybrid cloud approach for private cloud in the de-duplication mechanism, *2015 IEEE International Conference on Engineering and Technology (ICETECH)*, IEEE, pp. 1–3.

Tran, T., Pham, D.-T., Duong, Q. and Mai, A. (2019). An adaptive hash-based text deduplication for ads-b data-dependent trajectory clustering problem, *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, pp. 1–6.

Troncy, R., Rizzo, G., Jameson, A., Corcho, O., Plu, J., Palumbo, E., Hermida, J. C. B., Spirescu, A., Kuhn, K.-D., Barbu, C. et al. (2017). 3cixty: Building comprehensive knowledge bases for city exploration, *Journal of Web Semantics* **46**: 2–13.

Xia, W., Jiang, H., Feng, D. and Hua, Y. (2014). Similarity and locality based indexing for high performance data deduplication, *IEEE transactions on computers* **64**(4): 1162–1176.