Storage and Retrieval of Health Data in Distributed Environment

Thesis Submitted

by

Himadri Sekhar Roy

Doctor of Philosophy (Engineering)

Under the supervision of

Prof. Nandini Mukherjee

Department of

Computer Science and Engineering

Faculty Council of Engineering and Technology

Jadavpur University

Kolkata -700032, India

2025

Dedicated to

My little hero, Hrishit

Statement of Originality

I *Himadri Sekhar Roy* registered on 11th June 2018 do hereby declare that this thesis entitled "**Storage and Retrieval of Health Data in Distributed Environment**" contains literature survey and original research work done by the undersigned candidate as part of Doctoral studies.

All information in this thesis have been obtained and presented in accordance with existing academic rules and ethical conduct, I have fully cited and referred all materials and results that are original to this work.

I also declare that I have checked this thesis as per the policy on Anti Plagiarism, Jadavpur University, 2019 and the level of similarity as checked by iThenticate software is 7%

Himadri Sekhar Roy

Signature of Candidate
Date: 17/06/25

Nandini Mukherjee

Certified by Supervisor
(Signature with date and Official Seal)

Certificate from the Supervisor

*This is to certify that the thesis, entitled "**Storage and Retrieval of Health Data in Distributed Environment**" submitted by **Mr. Himadri Sekhar Roy** who got his name registered on 11th June 2018, for the award of Ph.D. (Engineering) degree of Jadavpur University, is absolutely based upon his own work under my supervision and that neither his thesis nor any part of the thesis has been submitted for any degree / diploma or any other academic award anywhere before.*

Nandini Mukherjee

Prof. Nandini Mukherjee
Professor
Department of Computer Science and Engineering
Jadavpur University
(Supervisor)

JADAVPUR UNIVERSITY

KOLKATA-700 032, INDIA

Index No: 27/18/E

1. **Title of the Thesis:**
   STORAGE AND RETRIEVAL OF HEALTH DATA IN DISTRIBUTED ENVIRONMENT

2. **Name, Designation and Institute of the Supervisor/s:**
   a. Prof. Nandini Mukherjee
      Department of Computer Science and Engineering
      Jadavpur University
      Kolkata-700 032

3. **List of Publications:**
   a. Refereed Journal:
      i. **Himadri Sekhar Ray,** Sunanda Bose, Nandini Mukherjee, Sarmistha Neogy and Samiran Chattopadhyay (2023). A cross-layer fragmentation approach to video streaming over mobile ad-hoc network using BATMAN-Adv. Multimedia Tools and Applications. 10.1007/s11042-023-16658-2.
      ii. **Himadri Sekhar Ray,** Md Mainuddin, Susmita Singha, Nandini Mukherjee (2024). A Survey of Big Data Tools for Remote Healthcare Services. The Journal of Supercomputing (Communicated).

   b. Refereed Conference Proceedings:

i. **Himadri Sekhar Ray,** Kausik Naguri, Poly Sen Sil and Nandini Mukherjee (2016). Comparative Study of Query Performance in a Remote Health Framework using Cassandra and Hadoop. 330-337. 10.5220/0005706803300337.

ii. Priyata Mukhopadhyay, **Himadri Sekhar Ray and** Nandini Mukherjee (2019). E-healthcare delivery solution. 595-600. 10.1109/COMSNETS.2019.8711429.

iii. **Himadri Sekhar Ray**, Swastik Mukherjee and Nandini Mukherjee, "Performance Enhancement in Big Data handling," 2020 International Conference on Contemporary Computing and Applications (IC3A), Lucknow, India, 2020, pp. 17-22, doi: 10.1109/IC3A48958.2020.233261.

iv. **Himadri Sekhar Ray,** Anurag Chakraborty, Radib Kar. (2021). Performance Enhancement in Big Data by Guided Map Reduce. In: Bhattacharya, M., Kharb, L., Chahal, D. (eds) Information, Communication and Computing Technology. ICICCT 2021. Communications in Computer and Information Science, vol 1417. Springer, Cham. https://doi.org/10.1007/978-3-030-88378-2_1

v. Gupta, A. **Himadri Sekhar Ray,** Nandini Mukherjee (2023). Strategic Network Model for Real-Time Video Streaming and Interactive Applications in IoT-MANET. In: Zhang, YD., Senjyu, T., So-In, C., Joshi, A. (eds) Smart Trends in Computing and Communications. Lecture Notes in Networks and Systems, vol 396. Springer, Singapore.

https://doi.org/10.1007/978-981-16-9967-2_26

c. **Doctoral Symposium**

i. Shreya, **Himadri Sekhar Ray** and Nandini Mukherjee (2022). Image Data Handling in a Multinode Environment Using MPI. In: Chaki, R., Chaki, N., Cortesi, A., Saeed, K. (eds) Advanced Computing and Systems for Security: Volume 13. Lecture Notes in Networks and Systems, vol 241. Springer, Singapore. https://doi.org/10.1007/978-981-16-4287-6_2

d. **Book Chapter**

i. Nandini Mukherjee, Sunanda Bose, **Himadri Sekhar Ray** (2020). A Framework for Delivering IoT Services with Virtual Sensors. 10.1201/9781003055976-8.

ii. **Himadri Sekhar Ray**, Sunanda Bose and Nandini Mukherjee (2022). Cloud-based Remote Healthcare Delivery and Its Impact on Society: A Case Study. In: Chaturvedi, M., Patel, P., Yadav, R. (eds) Recent Advancements in ICT Infrastructure and Applications. Studies in Infrastructure and Control. Springer, Singapore. https://doi.org/10.1007/978-981-19-2374-6_11

4. **List of Patents**: None
5. **List of Presentations in National / International Conferences**
   a. Mukhopadhyay, Priyata & **Himadri Sekhar Ray** and Nandini Mukherjee. (2019). E-healthcare delivery solution. 595-600. 10.1109/COMSNETS.2019.8711429.

b. **Himadri Sekhar Ray**, Swastik Mukherjee and Nandini Mukherjee, "Performance Enhancement in Big Data handling," 2020 International Conference on Contemporary Computing and Applications (IC3A), Lucknow, India, 2020, pp. 17-22, doi: 10.1109/IC3A48958.2020.233261.

c. **Himadri Sekhar Ray**, Anurag Chakraborty, Radib Kar. (2021). Performance Enhancement in Big Data by Guided Map Reduce. In: Bhattacharya, M., Kharb, L., Chahal, D. (eds) Information, Communication and Computing Technology. ICICCT 2021. Communications in Computer and Information Science, vol 1417. Springer, Cham. https://doi.org/10.1007/978-3-030-88378-2_1

d. Shreya, **Himadri Sekhar Ray,** Nandini Mukherjee. (2022). Image Data Handling in a Multinode Environment Using MPI. In: Chaki, R., Chaki, N., Cortesi, A., Saeed, K. (eds) Advanced Computing and Systems for Security: Volume 13. Lecture Notes in Networks and Systems, vol 241. Springer, Singapore. https://doi.org/10.1007/978-981-16-4287-6_2

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to all those who have played a significant role in the successful completion of my thesis. Although a thesis is known by the name of the author, there are many remarkable individuals without whose direct and or indirect help, support and encouragement, the thesis does not materialize. This journey involves countless cycles of exploration, inquiry, enlightenment, doubt, confusion, uncertainty and insistence. But it was a challenging experience, full of learning, and nice memories from the conference trips.

First and foremost, I am deeply thankful to my supervisor, Professor Nandini Mukherjee, whose unwavering guidance, expertise, and patience have been instrumental in shaping my research and academic growth. Her mentorship has been invaluable, and I am profoundly grateful for her dedication to my success. Without her suggestion, guidance, and constant effort, this work would never have been possible. I am always thankful to her for reviewing my thesis and papers. Her observations and comments helped me to establish the direction of my research and move forward to a deep investigation. I would like to thank her for allowing me to work under her supervision. her advice and help are invaluable to me and I'll remember these in all my life.

I would like to express my gratitude towards Professor. Sarmistha Neogy for the numerous technical discussions we had and their motivation during this period.

My thank goes to Professor Samiran Chattopadhyay, Vice Chancellor, Techno India University, for numerous technical discussion, and all the counselling, support and helpful feedback.

I gratefully acknowledge the research project "Remote Health: A Framework for Healthcare Services using Mobile and Sensor-Cloud Technologies" in the ITRA

research scheme under my supervisor, Prof. Nandini Mukherjee, Department of Computer Science and Engineering, Jadavpur University who was PI of the project. It helps me to find out my research challenge in a real domain. I would like to thank the School of Mobile Computing and Communication department, Jadavpur University, for supporting the lab work. I am also thankful towards the research project "Development of High-Speed MANET for Intelligent Collaboration" under Prof. Sarmistha Neogy, Department of Computer Science and Engineering, Jadavpur University for allowing me to serve as a junior research fellow.

I would also like to thank Mr. Rajib Bandyopadhyay, System Administrator, School of Mobile Computing & Communication, Jadavpur University for his support and due concern, making me aware of different aspects of research as and when enquired. He helped me a lot to set up the Labs at SMCC(School of Mobile Computing and Communication) to conduct various experiments for my thesis.

It gives me immense pleasure and it is a wonderful experience to work with the Department of Computer Science and Engineering, at Jadavpur University. The working environment is very nice and friendly. I would like to offer my sincere thanks to all of my colleagues. I also consider myself to be fortunate to have my friends Sunanda, Aamartya, Asif, Sriyanjana, and Atanu, elder brothers like Raju Mukherjee, Anil Routh of SMCC and many others, whose support, encouragement and willingness to listen made things bearable during the difficult time and delightful the rest of the times during the years.

My deepest appreciation and thanks to my parents Dr. Barun Kumar Roy, and Mrs. Sovana Roy and my in-laws who are the permanent source of love, encouragement and support. Their prayers are the reason behind my achievements. I am extremely grateful for their unconditional trust, timely encouragement, and endless patience. It was their love that raised me again when I got weary. I am greatly indebted to my wife, Rituparna for her tremendous and undying support without any complaint and sacrifice throughout these years. Words are not enough to thank my little son Hrishit, for his constant patience and enjoyment helping me to reduce my stress and finish my journey. His smile, love, and affection towards me always gave a strength and inspiration.

Last but not least, I am thankful to all my respected teachers from the school to the university, friends, relatives, and all the staff of the Department of Computer Science and Engineering, School of Mobile Computing and Communication, for their good wishes and support.

Himadri Sekhar Roy
Department of Computer Science and Engineering,
Jadavpur University,
Jadavpur,
Kolkata-700032,
India

Abstract

Providing primary health-care services remotely, in rural areas, particularly in developing countries, remains a significant challenge due to lack of inadequate infrastructure, high costs, lack of skilled personnel, and poor connectivity. Rural populations in these countries are mostly deprived of quality health-care services. To address this issue, it is required to implement innovative solutions made of modern technologies like mobile health (mHealth), cloud computing, sensor networks, and Big Data analytics. Several research have explored the use of these technologies in health-care domain to bridge the gaps between rural patients and health-care providers, ensuring that health-care is more accessible and affordable.

Kiosk-based remote health-care system is one important solution to overcome the issues. Kiosks are set up in rural villages, where local health-care providers use a touch-screen-based device and a mobile application to collect patients' vital signs and symptoms using a knowledge base which is created by doctors. The collected data is transmitted to cloud servers, allowing urban doctors to remotely access the information, prescribe medication, and recommend medical tests. In this research work, one such application is developed. Kiosks have been set up and the application has been deployed in multiple villages in West Bengal, India. A survey conducted after two years of operation showed that such systems could substantially improve health-care delivery in rural regions. The details of the implementation and the salient features of the application are presented in this thesis.

It has been observed that Internet connectivity is poor in the rural villages and in remote areas. In order to come up with a solution to this problem an SMS-based connectivity module is implemented to run the application at the time of problems with Internet connectivity. In addition to rural health-care services, the connectivity issue has also been dealt with for disaster sites where no network infrastructure exists. In such situations, the increasing proliferation of mobile devices has led to the exploration of Mobile Ad-hoc Networks (MANETs) as a framework for delivering real-time multimedia health-care services. MANETs allow transmission of multimedia data over a network of mobile devices without relying on a centralized infrastructured network. Despite challenges like network instability and packet loss, techniques such as cross-layered fragmentation have been used to improve real-time video and image transmission. This is particularly useful for carrying out rescue

operations in disaster sites and in remote health-care, where live video consultations between rural patients and urban doctors can enhance diagnosis and treatment.

The primary objective of this thesis is to explore the issues related to storage and management of health-care data. Considering that the health data has the properties of Big Data, the role of Big Data tools in health-care is crucial, particularly for storage and processing of the large amounts of data generated from various health-care devices along with the kiosks. Several Big Data solutions, including Hadoop, Cassandra, MongopDB, Neo4J, Hive, HBase have been examined for their applicability in health-care. Most of the tools designed for distributed storage and processing capabilities, to make them well-suited for handling massive health data such as Electronic Health Records (EHR). However, efficient mapping of health data onto these platforms requires careful consideration of data models and query structures. Comparative studies between Hadoop, Hive, HBase, Cassandra, MongoDB, Neo4j demonstrate their strengths and weaknesses in handling health-care queries, providing valuable insights into the selection of appropriate Big Data tools for health applications. While performing the experiment with the tools, some of the issues was recorded. To handle these issues, Node-Guided MapReduce concept was introduced.

The Node-Guided MapReduce (NGMR) framework represents a further advancement in Big Data processing through MapReduce. Traditional MapReduce implementations suffer from inefficiencies in execution of queries due to the non-selective involvement of nodes in the execution process, leading to increased execution time. NGMR addresses this issue through selective node selection and execution, thereby reducing query execution time and improves the overall performance of the system. The framework has been implemented and tested in small-scale environments using low-cost computational resources, offering a scalable solution for large health-care data systems.

The comparative analysis of various Big Data platforms such as Hadoop, HIVE, Cassandra, MongoDB, HBase, and NGMR has been done on health-care data, for health-care applications to determine the effectiveness of these tools in terms of performance in storing and query execution related to health data. The findings emphasize that, while NoSQL databases like Cassandra and MongoDB offer flexibility and scalability, the choice of platform should be driven by the specific needs of the health-care application, such as the volume of data and the type of queries being executed. On the other hand NGMR performs quite well in query processing.

Finally, an algorithm for automated node selection in NGMR has been proposed, which enables the system to efficiently guide queries to execute on an optimal set of nodes. To validate the performance of this node selection algorithm, a simulation set up using micro-services on AWS has been used.

The thesis concludes with the limitation of the research works presented in this thesis, outstanding issues and a direction towards the future work.

Contents

List of Figures

List of Tables

List of Abbreviations

ACMRV . . . Adaptive Cross-layer Mechanism for Real-time Video streaming

AD . . . . . Alzheimer's disease

AI . . . . . Artificial Intelligence

AODV . . . . . Ad hoc On-demand Distance Vector

ANM . . . . . Auxiliary Nurse Midwife

API . . . . . Application Programming Interface

ASHA . . . . . Accredited Social Health Activist

AWS . . . . . Amazon Web Services

BATMAN . . Better Alternative To Mobile ad-hoc Networks

BMI . . . . . Body Mass Index

CCPA . . . . . Central Consumer Protection Authority

CDMA . . . . Code-Division Multiple Access

COMIP . . . . Cluster-based Overlay and Fast Handoff Mobile IP System

CPU . . . . . Central Processing Unit

CRNT . . . . . Context Recognition Network Toolbox

CSV . . . . . Comma Separated Values

CT . . . . . Computed Tomography

DCT . . . . . Discreet Cosine Transform

DSR . . . . . Dynamic Source Routing

DSDV . . . . . Destination Sequenced Distance Vector

DVC . . . . . Distributed Video coding

ECG . . . . . Electrocardiogram

EHR . . . . . Electronic Health Records

FANET . . . . Flying ad-hoc Network

FPS . . . . . Frames per second

GCP . . . . . Google Cloud Platform

GPRS . . . . . General Packet Radio Service

GDPR . . . . . General Data Protection Regulation

GSM . . . . . Global System for Mobile Communication

HDFS . . . . . Hadoop Distributed File System

HIE . . . . . Health Information Exchange

HIPAA . . . . Health Insurance Portability and Accountability Act

HITECT . . . Health Information Technology for Economic and Clinical Health

ICT . . . . . Information and Communication Technology

IaaS . . . . . Infrastructure as a Service

IMP . . . . . Intelligent Medical Platform

IoMT . . . . . Internet of Medical Things

JSON . . . . . JavaScript Object Notation

KiORH . . . . Kiosk Operated Rural Health

LDAP . . . . . Lightweight Directory Access Protocol

LTE . . . . . Long Term Evolution

LRMA . . . . Long-Run Moving Average

MAC . . . . . Media Access Control

MDC . . . . . Multiple Description Coding

MHFW . . . . Ministry of Health and Family Welfare

MR . . . . . MapReduce

MRI . . . . . Magnetic Resonance Imaging

MT . . . . . Mobile Terminals

MTU . . . . . Maximum Transmission Unit

NAC . . . . . Neighbor-Aware Cluster-head

NGMR . . . . Node-Guided MapReduce

NoIR . . . . . No Infrared filter

OLTP . . . . . Online Transaction Processing

PaaS . . . . . Platform as a Service

PHC . . . . . Primary Health Care

PHP . . . . . Hypertext Preprocessor

PDC . . . . . Personal Digital Cellular

QoS . . . . . quality of service

REST . . . . . Representational State Transfer

RMP . . . . . Registered Medical Practitioner

RPM . . . . . Remote patient monitoring

RTT . . . . . Round trip time

SaaS . . . . . The services or software as a Service

SVC . . . . . Scalable Video Coding

SMS . . . . . Short Message Service

TDMA . . . . Time Division Multiple Access

TSV . . . . . Tab Separated Value

URL . . . . . Uniform Resource Locator

UAV . . . . . Unmanned Aerial Vehicles

VANET . . . . Vehicular ad-hoc Network

WHIMS . . . . The Wireless Health Incident Monitoring System

WLAN . . . . Wireless Local Area Network

WNIC . . . . . wireless network interface controller

WHO . . . . . World Health Organization

XaaS . . . . . Everything as a Service

YARN . . . . . Yet Another Resource Negotiator

The biggest value in Big Data for health-care will come from marrying the best technology with the best biology

— Arvind Krishna, *Chairman and CEO of IBM*

1 Introduction

Contents

1.1 Prelude

In recent years, the health-care industry has witnessed a transformative shift, driven by the integration of cutting-edge technology and the vast influx of data generated within its ecosystem. This paradigm shift has given rise to the concept of Big Data, revolutionizing the way health-care organizations manage, analyze, and utilize information. Big Data, characterized by its volume, velocity, variety, and complexity, has opened up new horizons for health-care providers, researchers, and policymakers alike. It offers unprecedented opportunities to enhance patient care,

improve clinical outcomes, streamline operations, and drive innovative research, all while ushering in a new era of data-driven health-care.

1.2 Background

Traditionally, health-care data was largely paper-based, making information retrieval and analysis a time-consuming and error-prone process. However, with the emergence of electronic health records (EHRs), diagnostic devices, wearables, and other digital health-care technologies, the industry now generates an overwhelming volume of data on a daily basis. This data encompasses patient records, clinical notes, medical images, genomic information, telemetry data, and more. The sheer magnitude of this information presents both an immense challenge and a tremendous opportunity.

Managing and storing the vast amount of health-care data is challenging, yet the potential benefits of Big Data in the health-care industry are transformative. By leveraging advanced analytics, machine learning algorithms, and artificial intelligence, health-care providers can analyze large datasets to identify patterns and correlations that were previously undetectable. Early detection of neurodegenerative diseases like Alzheimer's disease (AD) is possible with advanced algorithms rooted in deep learning and convolutional neural networks. These algorithms analyze a wide range of datasets, including medical records, genetic information, and imaging data, allowing the identification of subtle patterns. [61]

Predictive analytics, deep learning, anomaly detection, natural language processing, and personalized medicine have ushered in a proactive health-care era, leading to better patient treatment outcomes plans, reduced misdiagnosis, and cost-effective treatment procedures [99]. This enables early disease detection through predictive modeling, the development of personalized treatment plans tailored to individual patient needs, and overall improved patient outcomes. Additionally, Big Data supports population health management by identifying health trends, optimizing resource allocation, and informing evidence-based strategies for public health interventions, ultimately enhancing the efficiency and effectiveness of health-care delivery.

In this context, the development of robust and scalable storage and retrieval solutions for health-care Big Data is imperative. This involves the integration of cloud computing, distributed databases, data warehousing, and data lakes, among other technologies. It also necessitates the implementation of efficient data retrieval mechanisms to provide health-care professionals with quick access to relevant information when making critical decisions.

1.3 Characteristics of Big Data

Big Data contains a large amount of data that is not being processed by traditional the processing unit or data storage. It is used by many multinational companies(MNCs) to process the data and business of many organizations. Big Data is characterized by several key attributes that distinguish it from traditional data types. These characteristics are often referred to as the "3Vs" (Volume, Velocity, Variety) and have been expanded to include additional Vs (e.g., Veracity, Value, and Variability) to provide a more comprehensive understanding of Big Data. Here are the main characteristics of Big Data:

- **Volume:** The name Big Data itself is related to its enormous size. Big Data is a vast 'volume' of data generated from many sources daily, such as business processes, machines, social media platforms, networks, human interactions, and many more. Big Data technologies can handle large amounts of data. Big Data typically involves massive volumes of data. This can range from terabytes to petabytes or even exabytes of data. Managing, storing, and processing such vast amount of data require specialized infrastructure and technologies.

- **Velocity:** Velocity plays an important role compared to others. Velocity creates the speed by which the data is created in real time. Data in a Big Data environment is generated and collected rapidly, often in real-time or near-real-time.

Big Data velocity deals with the speed at which data flows from sources like application logs, business processes, networks, social media sites, sensors, mobile devices, etc. It involves managing the rate of incoming data, the pace of change, and sudden bursts of activity. The primary aspect of Big Data is to provide demanding data rapidly.

- **Variety:** Big Data includes a broad spectrum of data types, including structured, semi-structured, and unstructured data. In the past, data was gathered from traditional paper-based medical records, early electronic health records (EHRs), administrative data such as billing records and insurance claims, clinical trials, patient surveys, questionnaires, and disease registries. In the present, data collection has expanded significantly with advancements in technology. Electronic health records (EHRs) now store vast amount of digital patient information from hospitals and clinics, wearable devices and fitness trackers, genomic data from genetic testing, and digital medical images from technologies like MRIs and CT scans. Mobile health apps and social media platforms also generate health-related data. Additionally, Internet of Things (IoT) devices in health-care settings, such as smart beds and connected medical equipment, continuously collect data. These data are in various forms like text, images, videos, PDFs, Emails, audio, SM posts, photos, videos, etc.

  The data is categorized as below:

  - Structured data: Structured data is represented in a tabular form. Structured Data is stored in the relational database management system.

  - Semi-structured: In the case of semi-structured data, the schema is not appropriately defined, Some examples of semistructured data are emails, JSON, XML, CSV, TSV, files etc.

  - Unstructured Data: All the unstructured files, log files, audio files, and image files are included in the unstructured data.

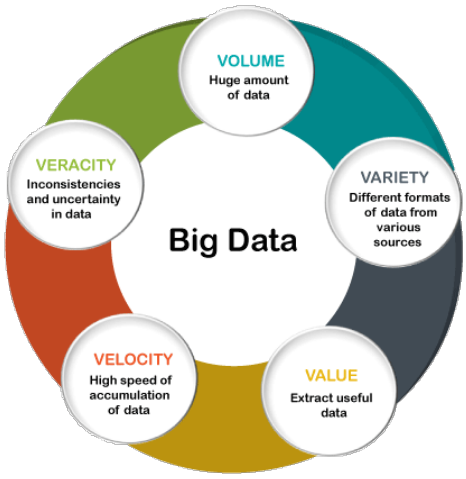

Figure 1.1: 5V's of Big Data

- Quasi-structured Data: Quasi-structured data is a mixture of structured and unstructured data. This type of data generally does not contain a fixed schema, but possesses some level of structure that makes it easier to process than completely unstructured data. Typical examples of quasi-structured data include web server logs, email headers, and sensor data. While logs and email headers are structured, the structure can vary between applications, making them quasi-structured rather than strictly structured.

- **Veracity:** Veracity means how much the data is reliable. It has many ways to filter or translate the data. Veracity is the process of being able to handle and manage data efficiently. Veracity refers to the accuracy and reliability of data. Big Data is also essential in business development, but often contains noise, errors, and inconsistencies, which can affect analysis and decision-making.

- **Value:** Value is an essential characteristic of Big Data. The primary purpose of collecting and analyzing Big Data is to extract valuable insights and knowledge. Big Data should contribute to improved decision-making, innovation, and business outcomes.

  There are some other characteristics of Big Data defined as

- **Variability:** Data in a Big Data environment may not have a consistent format or structure. Variability can be due to seasonal trends, market events, or other factors.

- **Complexity:** The complexity of Big Data goes beyond the data itself and includes complex relationships and patterns within the data. Analyzing Big Data often requires advanced analytic techniques, including machine learning and artificial intelligence.

- **Unpredictability:** Big Data may exhibit unpredictable patterns, making it challenging to anticipate future trends or behaviour.

1.4 Rquirements for Big Data

There are some requirements for Big Data

- **Accessibility:** Big Data should be easily accessible for analysis by authorized users. Integrating data from diverse sources can be challenging but is crucial for meaningful analysis.

- **Security and Privacy:** Protecting Big Data from unauthorized access and cyber threats is essential. Compliance with data privacy regulations is critical while handling sensitive information.

- **Scalability:** Big Data systems must be able to scale horizontally to accommodate growing data volumes. Scalability ensures that systems can handle increasing workloads without a significant drop in performance.

- **Distribution:** Big Data often resides in distributed environments, such as clusters or cloud platforms, to efficiently manage and process its large volumes. In these set ups, data is stored across multiple servers, enhancing both storage capacity and computational power. Clusters, which consist of interconnected computers working together as a single system, enable parallel processing and high availability, allowing tasks to be divided among multiple nodes for

faster analysis [84]. Cloud platforms offer scalable and flexible resources, dynamically adjusting storage and computation capabilities to match varying workloads and providing an array of tools for data analytics, machine learning, and real-time processing. By leveraging these distributed environments, organizations can achieve better performance, fault tolerance, and scalability, enabling them to derive valuable insights, improve decision-making, and drive innovation.

- **Parallel Processing:** Distributed processing frameworks like Hadoop facilitate parallel computation on large datasets by distributing data and computational tasks across clusters of computers [43]. Hadoop also includes a framework for parallel processing called MapReduce, which divides large tasks into smaller subtasks that can be executed independently across nodes, enabling efficient data processing and analysis. This distributed architecture allows tasks to be processed in parallel, leveraging the combined computational power of multiple nodes simultaneously.

- **Immediacy:** Many Big Data applications require real-time or near-real-time processing to make timely decisions.

1.5 Evolution of Big Data

In the last few decades, Big Data technology has gained significant growth. The evolution of Big Data has been a fascinating journey that has transformed based on the collection, storage, analysis, and derived insights from vast amounts of data. Here is a brief overview of its evolution:

- **Early Days:** Before the term "Big Data" was coined, organizations were already dealing with large volumes of data. However, data storage and processing technologies were limited. Relational databases like Oracle and SQL Server were popular, but they were unable to handle the massive datasets with variety.

- **Coining of the Term:** The term "Big Data" gained popularity in the early 2000s, with the rise of Internet-related service providers, which had to process and analyze massive amounts of data. In 2005, Roger Mougalas wrote an O'Reilly article titled "What is Web 2.0" that mentioned "data is the next Intel inside." This marked a turning point in the perception of data.

- **The 3Vs (Volume, Velocity, Variety):** Big Data was often characterised by the three Vs: Volume (large amounts of data), Velocity (the speed at which data is generated and processed), and Variety (the diversity of data types and sources). This framework helped conceptualizing the challenges of dealing with Big Data.

- **Hadoop:** Apache Hadoop, developed by Doug Cutting and Mike Cafarella, became the most popular Big Data storage and analytics tool. It introduced the concept of distributed processing of large datasets across clusters of computers. The Hadoop Distributed File System (HDFS) and MapReduce programming model allowed scalable and fault-tolerant data processing.

- **NoSQL Databases:** Traditional relational databases struggled to handle the variety and volume of Big Data. NoSQL databases like MongoDB, Cassandra, and HBase emerged as alternatives, offering schema flexibility and scalability for specific use cases.

- **In-Memory Computing:** Technologies like Apache Spark and in-memory databases (e.g., Redis, Memcached) gained popularity, allowing faster data processing and real-time analytics.

- **Cloud Computing:** Cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) made it easier to store and process Big Data by providing scalable and cost-effective infrastructure and services like Amazon S3 and AWS EMR.

- **Machine Learning and AI:** The integration of Big Data with machine learning and artificial intelligence allowed organizations to derive valuable insights, make predictions, and automate decision-making processes.

- **Data Lakes:** Data lakes, like Amazon S3 and Azure Data Lake Storage, became popular for storing and managing large volumes of raw and structured data, enabling data scientists and analysts to explore data more flexibly.

- **Streaming Data:** The rise of real-time data streams from sources like IoT devices and social media led to the development of stream processing frameworks such as Apache Kafka and Apache Flink, allowing immediate data analysis.

- **Data Governance and Privacy:** As data grew, so did concerns about data security, privacy, and governance. Regulations like GDPR and CCPA compelled organizations to be more responsible in handling and protecting data.

- **AI and Machine Learning Automation:** With the growth of Big Data, AI and machine learning technologies are increasingly automating data analysis, making it easier for organizations to extract valuable insights from large datasets.

- **Edge Computing:** Edge computing enhances the expandability of Big Data solutions by distributing the computational load across multiple edge devices. So the centralised processing power of cloud platforms increases drastically. This process enables more comprehensive and efficient data management to derive valuable insights from Big Data.

  Edge computing allows for the initial processing, filtering, and analysis of large datasets to occur closer to where the data is generated. This reduces the volume of data that needs to be transmitted to data centers, optimizing network efficiency and reducing costs.

The evolution of Big Data is ongoing, and it continues to shape various industries, including health-care, finance, marketing, and more, by enabling data-driven decision-making and innovation. The future of Big Data will likely involve advancements in data storage, processing, analytics, and ethical considerations surrounding data usage.

1.6 Application of Big Data in health-care

In recent years, significant attention has been given to Big Data due to its potential to revolutionize various industries, including health-care. Both public and private organizations utilize Big Data techniques to generate, store, and analyze data, ultimately enhancing their services. In health-care, Big Data sources include electronic health records (EHRs), lab reports, patient portals, wearable devices, smartphones, and search engine data, among others. Health-care Big Data encompasses patient, hospital, doctor, and medical device information, analyzed to improve disease treatment, reduce costs, and prevent epidemics.

Previously, collecting health data was labor-intensive and time-consuming, but modern technologies have streamlined this process. The primary motivation for using Big Data tools in health-care is to predict and address health issues early, reduce treatment costs, engage patients effectively, raise disease awareness, and improve medical inventory management. Key applications of Big Data in health-care include:

- **Real-time patient monitoring:** Big Data analytics enable hospital staff to work more efficiently through sensors embedded in beds that continuously monitor patient vitals such as heart rate, blood pressure, and respiratory rate. Sudden changes trigger real-time notifications to hospital staff, facilitating timely medical intervention.

- **Hospital administration:** Big Data helps management to optimize staff allocation and other resources, thereby reducing costs.

- **Enhancing patient engagement:** Wearable devices that track health metrics like blood pressure and heart rate can be integrated with other data to provide physicians with comprehensive patient information, reducing unnecessary doctor visits.

- **Telemedicine:** Telemedicine delivers remote treatment, including patient monitoring, medical education, and even remote surgery using robots.

- **Fraud prevention and detection:** Big Data applications help to detect and prevent fraud in health-care, such as incorrect dosages or medicines. Insurance companies also use Big Data to reduce fraudulent claims.

- **Medical image analysis:** Big Data aids in developing better algorithms for image analysis, helping radiologists store, manage, and examine medical images more efficiently, reducing manual efforts.

1.7 Application of IoMT

The Internet of Things (IoT) connects electronic devices, apps, and sensors to exchange data and solve various problems. In health-care, this concept evolves into the Internet of Medical Things (IoMT), facilitating health monitoring and treatment. Key applications of IoMT include:

- **Pandemics:** IoMT-based robots gained popularity during the COVID-19 pandemic. Artificial intelligence and data mining helped to create predictive models to manage the crisis.

- **Health vitals monitoring:** IoMT supports virtual doctors and health-care monitoring systems, connecting physical doctors only when necessary and prescribing medication based on biometric data.

- **Diabetic monitoring:** IoT offers blood glucose monitoring systems, with methods ranging from invasive to non-invasive.

- **Robotic surgery:** IoMT drives advancements in robotic surgery, enhancing precision and reducing manual intervention.
- **Ingestible sensors:** These sensors, embedded in digital tablets, monitor internal health metrics like blood pH and gastric enzymes, aiding diagnostics and treatment.
- **Pacemakers:** IoT-enabled wireless pacemakers notify doctors of a patient's condition, improving heart health management.

1.8 Challenges faced in resource-poor areas

Medical data is highly sensitive, and any error can significantly impact analysis outcomes. Key challenges for Big Data in health-care include:

- **Core decision sensitivity:** Big Data analytics require precise and timely patient information, particularly for critical health issues.
- **Data conversion issues:** Health-care data comes in various formats and from multiple sources, complicating storage and analysis.
- **Data availability and quality:** Successful health-care data analysis depends on data availability and quality, requiring technology experts to standardize data formats.
- **Laws and regulations:** health-care regulations pose challenges for implementing Big Data analytics, as diagnostic and treatment decisions have high stakes.
- **Infrastructure and expertise availability:** A shortage of professionals and experts hinders the implementation of Big Data tools in hospitals.
- **Sustainability of health-care models:** Financial constraints limit the adoption of Big Data technologies in health-care.

- **Ethics and privacy:** Data privacy and ethical concerns restrict the use of health data for Big Data analytics.
- **Integration of proper tools:** Optimizing the health-care system requires seamless integration of diverse workflows and electronic health records.
- **Lack of trust:** Trust issues regarding the use of health data hinder the adoption of Big Data analytics.

IoMT, while revolutionary, also faces challenges:

- **Data security:** Protecting patient demographics and vitals in IoT systems is critical, requiring robust server protection technologies.
- **Protocol integration:** Different devices and health sensors use various protocols, complicating data collection and requiring compliance with standards like HIPAA and HITECH.
- **Data overload:** Excessive data can reduce accuracy and overwhelm health professionals, necessitating efficient data management solutions.
- **Cost:** The high cost of IoT technologies poses a challenge, making it essential to reduce costs for broader accessibility.

1.9 Research motivation and objective

The current scenario reveals that the use of the Internet and telecommunication networks of India are increasing day by day as smartphones and tablets are becoming cheaper. A mobile-based solution can be more effective in remote health-care systems using the Internet and cloud technologies.

The implementation of primary health-care services using mobile and cloud technology generates huge unstructured data. This heterogeneous data is stored in the cloud in a distributed environment and needs to access mobile devices. So, storage and retrieval of health data are two challenging issues for the implementation of primary health-care services in rural areas. This thesis addresses these main

three issues, i.e. providing proper rural primary health-care services remotely using mobile and cloud technology, efficient storage and retrieval of health data, and transfer the text and multimedia data when Internet connectivity is not available.

The research targets the following objectives:

- **To provide primary health-care service schemes for rural areas:**
  The primary goal is to develop and implement effective health-care service schemes that specifically address the challenges faced in rural and hard-to-reach areas.

- **To propose a scheme for efficient health and multimedia data transmission in non-traditional ways:**
  This objective aims at developing a novel scheme for transmission of health data and multimedia content (such as medical imaging or real-time video consultations) in resource-constrained environments, including areas with limited Internet connectivity.

- **To develop technique which efficiently store and retrieve large amount of health data in distributed storage:**
  As health-care systems generate huge amount of heterogeneous data, efficient storage and retrieval become critical. This objective focuses on designing a distributed storage system that can handle the large volume of data generated by health-care applications.

- **To enhance the scalability and performance of health data management systems:**
  This objective focuses on enhancing the scalability and performance of the entire health data management ecosystem. The system should be capable of accommodating larger datasets, supporting more users, and handling complex queries without sacrificing performance.

1.10 Contributions

The focus of the thesis is to address the challenges faced in rural health-care services using cloud and mobile computing technologies and findings and find suitable technologies to store and efficient retrieval of heterogenous health data. The main contributions of the thesis are described bellow:

- **Development of an ICT-based Framework for the KiORH Application for Remote health-care**

  The research presented in the thesis involves the creation of an ICT-based (Information and Communication Technology) framework for a rural health-care application. This application and the designed framework enhances remote health-care facilities for hard-to-reach population by leveraging a combination of sensors, mobile technology, and cloud-based systems. The framework facilitates real-time health data monitoring, collection, and transmission. Sensors capture vital patient data, which is securely transmitted and stored in cloud storage and databases. Additionally, health-care providers in urban areas can remotely monitor patient health and make timely decisions. The efficiency and scalability of the system make it suitable for remote health-care delivery and emergency response scenarios in primary health-care.

  Furthermore, an SMS (Short Message Service) based transmission scheme for text and a Cross-Layer Video Fragmentation and Transmission scheme for video transmission over a mobile ad-hoc network (MANET) has been developed. These two schemes are particularly significant for applications where there is limited Internet connectivity or in disaster management or in where, the condition of medical facilities is deplorable. The real-time testbed implementation of these two schemes demonstrates their potential to maintain quality video and data transmission, even in network-constrained environments.

- **Comparative Analysis of Distributed Storage Tools**

  In the domain of distributed data storage and retrieval, I conducted a

comparative analysis of widely used tools such as Hadoop, Hive, HBase, Cassandra, MongoDB, and Neo4j for health-care data. These tools were evaluated for their suitability in handling the unique challenges posed by health-care data, specifically for performance in both structured and unstructured environments. By testing each tool under a similar environment, the pros and cons were evaluated in terms of data throughput, query performance, and storage efficiency. This analysis provides valuable insights for future implementations of distributed storage systems in health-care environments, offering a clear understanding of the trade-offs involved in selecting a storage technology.

- **Proposal of a Node-Guided MapReduce Scheme**

  To address the inefficiencies in traditional NoSQL and MapReduce models when applied to big health-care datasets, I proposed a novel Node-Guided MapReduce scheme (NGMR). The scheme enhances the performance of MapReduce by introducing a novel approach that performs computational tasks only on guided nodes. This approach ensures optimized data processing, leading to faster results and more efficient use of computational resources. The framework was implemented and compared with other NoSQL databases and traditional MapReduce schemes, demonstrating a significant reduction in execution time while maintaining the integrity and accuracy of the results.

- **Algorithm for Automated Node Selection in Node-Guided MapReduce**

  Further research was conducted on the Node-Guided MapReduce scheme for the Automated Node Selection process. This algorithm selects the most suitable nodes for an incoming query, based on real-time factors such as node availability, processing power, network latency, etc. The algorithm enhances the overall efficiency of distributed systems by selecting nodes that store the required data, leading to faster query resolution and optimized resource utilization.

1.11 Organization of this thesis

The chapters of this thesis are organized as follows:

Chapter 2 discusses the background and literature review of eHealth, digital health, and mHealth. Along with addressing the challenges of mobile health applications, this chapter also describes an ICT-based framework for remote health-care and the challenges of cloud-based health-care services in a Big Data environment.

Chapter 3 introduces an ICT-based framework and a "Kiosk-Based Remote health-care" solution for rural primary health-care. This chapter also explains the usefulness of the application in rural areas.

Chapter 4 proposes approach to video transmission over a mobile ad-hoc network and presents the relevant algorithms. The methodologies for storage and retrieval of videos to improve the efficiency of video streaming are also discussed. In addition to video transmission, this chapter briefly discusses SMS-based remote health-care services, establishing a communication channel between rural patients at health kiosks and doctors consulting from urban centers through SMS, using data encryption and compression.

Chapter 5 presents an overview of various Big Data tools and their impact on the health-care system. This chapter also describes the challenges associated with the use of Big Data tools for the storage and retrieval of health data.

Chapter 6 provides an overview of the proposed Node-Guided MapReduce framework, along with a presentation of the key phases and features of the system. This chapter also discusses the setup of the proposed framework using a testbed.

Chapter 7 offers a comparative analysis of various Big Data tools and examines their performance. This chapter also describes the data models and different file structures used for each tool.

Chapter 8 proposes a node selection algorithm for the Node-Guided MapReduce framework. This chapter also provides implementation details of node selection algorithm using microservices on Amazon AWS.

Chapter 9 concludes with an overview of the thesis, its significant results, and discussions. It also addresses outstanding issues, limitations, and directions for future work.

eHealth is the future of health-care. It has the potential to transform health-care delivery, improve patient outcomes, and reduce costs

— Dr. Tom Insel, Former Director of the National Institute of Mental Health

2

eHealth, digital health, mobile health

Contents

2.1 Prelude

eHealth encompasses a broad spectrum of electronic tools and systems that contribute to health-care management. From Electronic Health Records (EHRs) that streamline patient data to telemedicine platforms facilitating remote consultations, ehealth initiatives aim to create a seamless and interconnected health-care ecosystem. The digitisation of health records not only improves data accuracy, but also allows real-time communication among health-care providers for studying collaborative

and informed decision-making.

On the other hand, mHealth uses mobile devices to bring health-care directly into the hands of individuals. With the widespread adoption of smartphones and smartwatches, mHealth applications empower users to monitor their health, receive personalized wellness insights, and engage in virtual consultations with health-care professionals. The portability and convenience of mHealth make it a powerful tool for preventive care, chronic disease management, primary health-care services, and health education.

Together, eHealth, mHealth, and digital health synchronize to redefine patient care by integrating electronic health records with mobile applications, thereby ensuring that health information is not only accessible but also actively engages individuals in their well-being. With the collaborative nature of these technologies, facilities have a more patient-centric approach, where health-care becomes proactively personalised and increasingly democratized. As the world continues to embrace the digital era, eHealth, mHealth and digital health stand as pivotal pillars, driving a paradigm shift towards more connected efficient and patient-centric health-care experiences.

The rapid increase in the usage of mobile devices, such as mobile phones, and tablets and the innovation of newer medical sensor devices with standard communication interfaces facilitate access to health data in real-time scenarios. Using these technologies, a proper medical service can be developed at the primary level in the rural areas of India. This thesis describes a real-time application based on sensor, mobile, and cloud technologies that provides an integrated environment for patients, caregivers, and remotely located medical professionals to interact meaningfully and create a treatment plan for delivering basic health-care services to rural patients.

The next few sections aims to provide insights into eHealth, mHealth and cloud-based remote health-care delivery and their impacts on the society.

2.2 eHealth

In the current digital age, the convergence of technology and health-care has given rise to a transformative phenomenon known as electronic health or eHealth. eHealth encompasses the use of ICT to improve the delivery and management of health-care services. This innovative approach leverages digital tools to enhance health-care efficiency, accessibility, and overall patient outcomes. As the complexities of modern health-care are navigated, hope is offered by eHealth, with a promise to revolutionize the way medical care is perceived, accessed, and received. This multifaceted concept encompasses a broad spectrum of applications, including electronic health records (EHRs), telehealth, Health Information Exchange (HIE) and mobile health (mHealth).

- **Electronic Health Records (EHRs):** One of the cornerstones of eHealth is the adoption of Electronic Health Records (EHRs), which replaces traditional paper-based medical records by providing a centralized or distributed digital repository of all health-related information, from patient demographics to doctor availability schedules. EHRs facilitate seamless communication among health-care professionals, reduce errors associated with manual record-keeping, and enhance the overall quality of care. Moreover, EHRs enable patients to access their health information, promoting informed decision-making and health-care engagement.

- **Telemedicine and Telehealth:** eHealth extends health-care beyond the barriers of the traditional approaches through telemedicine and telehealth services. Telemedicine platforms enable patients to consult with health-care professionals remotely using ICT, video consultations, remote monitoring, and virtual follow-ups by breaking down geographical barriers and improving accessibility, particularly for individuals in rural or underserved areas.

- **Health Information Exchange (HIE):** Interoperability is a key element of eHealth, facilitated by Health Information Exchange. This allows the seamless

sharing of patient information among different health-care entities, promoting collaborative and patient-centred care. HIE enhances communication between hospitals, clinics, laboratories, and pharmacies, fostering a more integrated and holistic approach to health-care.

- **Mobile Health (mHealth):** The ubiquity of smartphones has paved the way for mobile health applications, offering a myriad of tools for health monitoring, wellness promotion, self-monitoring and disease management. From fitness trackers to medication reminders, mHealth empowers individuals to actively participate in their health-care journey. This not only improves patient engagement but also contributes to preventive care strategies for chronic and elderly people.

2.3 mHealth

In an era dominated by technological advancements, Mobile Health (mHealth) has emerged as a revolutionary force, transforming the landscape of health-care delivery. mHealth refers to the application of mobile technology, such as smartphones and tablets, to support and enhance medical and public health practices. With the global proliferation of mobile devices, mHealth has become a powerful tool in promoting accessibility, efficiency, and patient engagement within the health-care ecosystem. The primary strength of mHealth lies in its ability to transcend geographical barriers and bring health-care services to the fingertips of individuals worldwide using ICT. In remote or underserved areas where traditional health-care infrastructure may be lacking, mHealth serves as a bridge, connecting patients with health-care professionals through mobile applications. This accessibility is particularly crucial in emergencies, where timely intervention can be a matter of life and death. Some benefits of the mHealth are:

- **Telemedicine and Remote Consultations:** Telemedicine through mobile devices enables remote consultations between health-care providers and patients. Through video calls, voice messages, or text-based communication,

individuals can seek medical advice, receive diagnoses, and even obtain prescriptions without the need for physical visits to health-care facilities. This not only improves access to health-care, but also reduces the cost of traditional health-care systems.

- **Health Monitoring:** mHealth incorporates wearable devices and sensors that allow to monitor the health of individuals in real time. From fitness trackers measuring physical activities to smartwatches recording vital signs, and sending those vitals to connected smartphones - all such wearable technologies empower individuals to take an active role in managing their well-being. This proactive approach to health monitoring can lead to early detection of potential issues and the implementation of preventive measures, fostering a culture of personalized health-care.

- **Elderly and Patient Care through Remote Monitoring:** Elderly care centers are increasingly exploring remote monitoring techniques as a viable and cost-effective option [22]. The elderly persons are particularly susceptible to emergencies such as wandering off from care homes, falls, and abnormal changes in vital signs. Therefore, there is a critical need for continuous and reliable monitoring of residents in these centers, preferably with real-time built-in alerting mechanisms for emergencies. Remote patient monitoring (RPM) of physiological measurements offers the potential to deliver high-quality care to elderly and chronically or acutely ill individuals in their home environments, while optimizing the use of health-care resources. Modern sensors can measure vital signs such as blood pressure, body temperature, blood sugar levels, heart rate, respiration rate, ECG readings, and brain activity.

- **Chronic Disease Management:** For individuals living with chronic conditions, mHealth plays a pivotal role in daily management. Mobile applications offer tools for medication adherence, symptom tracking, and lifestyle management. Patients with diabetes, for example, can use mHealth apps to monitor blood glucose levels and receive personalized recommendations for diet and

exercise. This not only improves the quality of life for those with chronic conditions, but also reduces the frequency of hospital visits.

2.4 The current state of the art in electronic and mobile health

eHealth and mHealth applications are significantly increasing around the world. Mobile medical data capturing, processing and computing systems for continuous monitoring and control of patients' health conditions are proposed by Banerjee et.al [30]. Numerous mobile applications are available to maintain a healthy life and public awareness [66], [37], [25], [67]. A framework for continuous acquisition of patient condition data is presented by the authors of [86]. Onini et. al. [87] conducted research studies and evaluation of on mHealth and eHealth apps for evaluation in Caucasian regions, Georgia. According to the surveys [70], 62.5%-86.5% of patients and 70%-90% of physicians ranked mHealth and eHealth services as cost- and time-efficient methods. Android smartphones with LRMA (Long-Run Moving Average) prerecorded software were used to evaluate cardiac arrhythmia in 73 patients [44]. The diagnosis of 11% of research participants was clarified based on Mobile Telemonitoring (m-TM) methods. The cost of the diagnosis process was reduced 2.7 times than average expenditures at hospital cardiac wards ($p<0.005$).

2.5 Use of smartphones

In the last two decades, mobile phones have seen the highest technological growth in the digital era. The mobile phone started its journey in the late 80s with only the voice facility and at that time it was called the first generation or 1G. After years, with the invention of global systems for mobile communication technology like GSM, TDMA, PDC, and CDMA, mobile phones entered into the second generation or 2G system in digital communication. Then the general pocket radio service GPRS was introduced. On the other side, wireless LAN and mobile AdHoc Networks were introduced in the mobile system to send that data at high speed. Presently in India,

in most parts of the country, wireless coverage is available at a very cheap cost. The mobile system adopts the information and communication technology. After the introduction of 4G and 5G communication, mobile smartphones can be used for many state-of-the-art applications, including the Internet of Things (IoT).

Smartphones especially are a type of mobile phones which incorporate the function of a palmtop, laptop, personal digital assistant or similar devices along with the audio and video player, calculator or payment devices and many more. In addition to that, a mobile phone or a smartphone contains many digital intelligent functionalities. So it can be said that a smartphone or a powerful devices combines the conventional function of a mobile phone with advanced computing capabilities enabling users to access multiple installed software applications in it.

The use of smartphones, tablets or mobile devices in health-care is commonly referred to as mobile health or mHealth. According to the recent reviews, mobile applications were used by 5000 million smartphone users in 2015, and the same has grown to 120% of all users in 2023 [111] in India only. With an estimation of over 4000 mobile health applications, almost 250000 individuals have downloaded at least one application [77]. The following applications of mHealth are deployed in India.s

- Gramin Health-care was founded to provide affordable primary health-care and specialist care in India's rural and poorest regions [63].
- iKure Health Monitoring kiosks were set up in rural areas of West Bengal, India where health-care infrastructure is negligent. The kiosks are equipped with the wireless health incident monitoring system [21].
- e-Mitra is a kiosk-based health-care delivery solution for rural areas, launched by the government of Rajasthan, India. Through this application, an expert doctor across the globe attends the patients in rural areas.
- There are numerous well-known and popular mobile-based healthcare applications, such as Dr Lal PathLabs, Practo [1], Credihealth [2], Tweet2Health,

MedicExpress, Curofy [3], Healthonphone, Netmeds [4], Symptomate Symptom Checker [5], MDCalc [6], Prognosis: Your Diagnosis [7] which function effectively in areas with sufficient network bandwidth.".

G.Spina et al. have developed an Android open-source smartphone framework CRNTC+ for sensor data acquisition, signal processing, pattern analysis, interaction and feedback, based on the Context Recognition Network Toolbox (CRNT). CRNTC+ extends the original CRNT by providing components to read smartphone and external sensor data, supporting annotations, and various output components [110]. Nowadays, A growing number of elderly citizens living alone, are prone to get involved in using smartphones and researchers are trying to find how the smartphone can support health-care and independent living. Zhou et al. [125] developed an effective and convenient activity detection system which plays an important role in health-care services. The application uses a system of learning living status using a finger-worn device which detects daily activities and shares information with the user's smartphone. As multiple processing is executed in a smartphone device, it is normal that the battery gets drained easily. All wearable devices indeed use the personal smartphone device as a gateway, and thus it is a challenge to keep the battery un-drained. Smartphones have the downside of low battery power and are unable to transmit their data to the medical personnel when the patient is on the move and away from the fixed gateway at the smart home/smart clinic. Sigwele et. al., in their paper [108], describe an intelligent and energy efficient 5G-based smartphone gateway for health-care smart devices (IEE5GG). In IEE5GG, the 5G architecture is adopted and the patient's smartphone is used as a gateway where multiple smart devices are connected, e.g. via Bluetooth. To save energy, requests to the smartphone can either be executed on the smartphone gateway or offloaded and executed in the Mobile Edge Computing (MEC) cloud at proximity to the smartphone in the 5G Base Station (BS) Central Unit (gNB-CU) while considering the transmission power, Quality of Service (QoS), smartphone battery level and Central Processing Unit (CPU) load.

The integration of smartphones into health-care represents a powerful synergy between technology and well-being. From promoting individual health management to facilitating advanced health-care delivery models, smartphones have become catalysts for positive change in the health-care landscape. As the field of mHealth continues to evolve, smartphones will likely play an increasingly central role in shaping the future of health-care.

2.6 Mobile application for health-care

In India, almost a billion users use mobile phones and approximately three-fourth of the population have smartphones [8]. In such scenarios, there is a huge possibility of using mobile apps in remote health-care like patient management, medicine management, health record storage, doctor availability calculation, prediction etc. This kind of task increases efficiency, saves times, reduces cost and easily analyses the diseases leading to more research on the diseases. In the last five years, many health apps have been developed by various software companies to fulfil the basic needs of health-care services. This kind of health applications can be broadly categorised into some classes according to the users of the app in the medical field. Some of these are:

- information gathering: in this category, the health app is used to reference medical drugs. Some of these apps are 1MG, Pharmeasy, practo, Apollo Health etc.

- Online / Remote Consultation: in this category, users use such apps to consult doctors through audio, video, textual chat etc. Some of the examples are Practo, Apollo 24/7 etc.

- Clinical decision-making system: in this category, the apps are used for making clinical decisions. Some of the examples are visual DX, prognosis etc

- Patient monitoring: patient monitoring apps like iCare, Health Track, Jotform, Dexcom, Senseonics, Medtronic etc fall in this category.

- Advance Alert notification: There are some apps like Vax, CDC Vaccine etc which are used as alerts.

2.7 Challenges of mobile applications

Mobile apps for health-care have the potential to revolutionize health-care delivery, but several challenges and issues need to be carefully addressed. Here are some key issues associated with mobile apps in health-care:

- In remote health-care rural people are not much conversant in using mobile apps on their smartphone. In some places in rural villages, mobile internet connectivity is not available. So in this case mobile phones are not capable of providing health-care services.

- Sometimes users or not willing to use health-care apps due to their complex user interfaces. If the medical apps fail to address that user's convenience then unfortunately they bring more trouble for the rural people. Thus in the case of remote health-care, clean and easy interfaces are much more required.

- In many cases, it is noticed that the developer cannot guided by the doctors while creating the app. Clinical advice is much required for app development as well as for collecting the clinical information of patients. Unfortunately, most of the medical apps fail due to missing the health condition of the patient and inappropriate treatment procedures.

- Mobile health apps often handle sensitive personal health information. If those are not adequately secured, then they can become a target for cyber attacks with compromising patient privacy.

- Most of the mobile apps or not in compliance with the Health Insurance Portability and Accountability Act HIPAA.

- Many mobile health apps operate in isolation, leading to challenges in interoperability with the other health-care system. Lack of standardised

data format and communication protocol block the seamless information exchange between different platforms and health-care providers.

- Sometimes, integration with the health app with the existing manual EHR system becomes very complex and requires adherence to industry standards.
- The reliability of health-related information provided by different apps may vary. This can raise a concern about the accuracy and trust the needs of medical apps
- Due to the rapid evolution of eHealth and mHealth technologies the regulatory bodies are struggling to keep pace. Alongside different countries have different ethical concerns. Not all apps are made to fulfil those ethical concern in all countries.

2.8 Cloud based health-care

Cloud-based health-care, often referred to as *health cloud*, or, *Health-care as a Service (HaaS)*, represents a transformative shift in the way health-care data is stored, managed, and accessed. This innovative approach leverages cloud computing technologies to store electronic health records (EHRs), medical images, and other health-care-related information securely in off-site servers. Cloud-based health-care systems offer several advantages, including enhanced accessibility, scalability, and collaboration among health-care providers. With data stored in the cloud, authorized users can access patient information from anywhere with an Internet connection, facilitating seamless care coordination. The scalability of cloud infrastructure allows health-care organizations to adapt to changing storage needs and accommodate the increasing volume of medical data. Moreover, the cloud enables the integration of advanced analytics, artificial intelligence, and machine learning applications to derive meaningful insights from vast datasets, ultimately contributing to improved patient outcomes and efficiency of health-care delivery. Despite the benefits, concerns related to data security, privacy, and regulatory compliance necessitate robust measures to ensure the confidentiality and integrity of sensitive health information in the cloud.

Cloud computing provides computational services over the Internet to the public. The services or software as a Service(SaaS), Platform as a Service(PaaS), Infrastructure as a Service(IaaS), and Everything as a Service(XaaS). The SaaS is the cloud service that provides the end user swith web-based software applications. The end users should have conventional web browsers or specific client software to avail services. In PaaS cloud service, the end-users can choose or develop their operating platform to enable the computing environment. IaaS is a cloud service where the users have full access to the cloud infrastructure. The users can implement their platforms and applications there. XaaS is the service where the end-user can get the above-said services together. Multiple vendors are providing cloud services. Amazon EC2, IBM Cloud, Microsoft's Azure, Google Cloud, Salesforce etc are some of the leading cloud service providers.

The cloud computing deployment models available for end users are private, community, public, and hybrid [116].

- In a private deployment model, the cloud infrastructure is provisioned for individual users or sometimes it can be for an organization or a company.
- The public deployment model is used for the general public users and is operated by some cloud service providers.
- The hybrid cloud deployment model combines two or more deployment models, allowing them to operate independently while managing different tasks.
- **Benefits of Cloud-Based Health-care:** Cloud infrastructure scales readily to accommodate growing data volumes and user bases, offering a cost-effective solution. As cloud infrastructure deployed in a distributed environment, it offers robust security measures and disaster recovery plans, ensuring data integrity and accessibility even during emergencies. Patients can access their health records, gathered from wearable wireless sensors [72] and data through secure portals, fostering informed decision-making and engagement in their treatment plans. On the other hand, cloud-based solutions eliminate the need

for expensive on-premise hardware and IT staff, leading to significant cost savings in the long run [35].

- **Challenges and Considerations:** Stringent data privacy regulations like HIPAA need to be adhered to, requiring robust data governance frameworks and patient consent mechanisms. As health-care systems require standardized data formats and open APIs, they need to ensure seamless data exchange across different cloud platforms, which can be a complex thing to maintain. As all the services are connected using active Internet connectivity, reliable Internet access and digital literacy training are crucial for equitable access to cloud-based health-care. For secure access to the data, a security module is needed to enhance the safety and privacy of patients. To improve security in the eHealth system, biometric confirmation systems can be implemented to prevent unauthorized access and mitigate the risks associated with forgotten passwords [62].

2.9 ICT based framework

In the contemporary landscape of health-care, the integration of Information and Communication Technology (ICT) is revolutionizing health-care by providing innovative tools and solutions to enhance patient care, improved health-care delivery, and optimize resource utilization. The ICT-based framework for health-care is a comprehensive system that leverages advanced technologies to streamline processes, enhance communication, and improve overall health-care outcomes. An ICT-based framework for health-care serves as a comprehensive roadmap for integrating ICT into various aspects of the health-care system. In the digital era, telecommunication networks are spread and Internet-based applications can be built easily on top of it due to its widespread availability among a large section of people through their ICT devices. In remote health-care systems, mobile phones and tablets can be used as ICT devices more efficiently along with cloud computing. The disease information, and symptoms with guidelines are stored in the cloud or server as

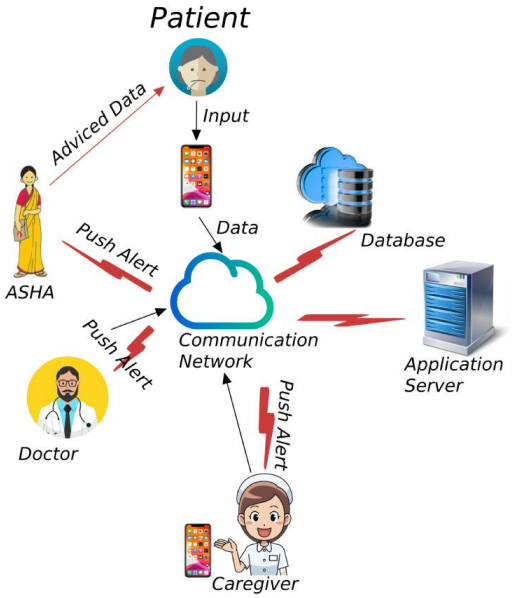

Figure 2.1: ICT-based framework for smartphone assist remote health-care

Big Data, so that they can be accessed easily through ICT-enabled devices. In India, there are many areas in primary health-care where mortality rate [9] is very high due to limited services, infrastructure, disconnected communications and fewer human health-care resources.

An ICT-based framework with a collaboration of mobile and cloud computing is shown in Figure 2.1. Two servers are depicted in the framework. One is the application server, the other is a database server. The scheme is implemented in primary health-care. In this framework, an app is installed on the tab or mobile phones connected to the Internet. After collecting the vitals of the patient through this app, the app transmits the vitals to the server. The application-related data come from the application server. For storing the patient records, the database server is used.

Whenever a new record is stored, the application sends a notification through the server to the application which is installed at the doctor's end. Doctor can check and prescribe the medicine, treatment plan, and medical tests using the application which is installed at the doctor's end. The server sends the notification to the application which is installed in the mobile phones or the tablet at the patient's

end. The patient can check the prescribed medication, tests, and treatment plans by opening the installed mobile application.

2.10 Challenges of health-care servies

The health-care system faces many challenges including more health-care costs, patient safety, overtreatment and failure to adopt best practices for health-care. This section describes the challenges [49]

- **Cost and Quality of health services:** Major challenges for improving health-care services in rural areas is how to maintain a balance between the cost of health services and their quality. Providing advanced health-care services at low cost is itself a challenge.

- **Human resources:** The shortage of health professionals such as doctors, nurses, and lab assistants is another challenge. The reasons for these include, poor living conditions, bad working conditions, excessive responsibilities, absence of opportunities, shortage of health workers, and lackof schooling and professional development. So it is hard for a doctor to efficiently monitor such the large number of patients in rural areas.

- **Affordability and Acceptability:** Backward economic conditions are the biggest problem in rural areas. Numerous people face constraints in accessing to health-care services, because of distance to hospital, lack of money for transport, fewer transportation devices, and poor health-care services. Lots of NGOs and governments from state and country levels are trying to strengthen their financial system to reduce the cost of health-care services.

- **Inequity in Health-care:** Health-care services vary based on the different perspectives of socio-stratification. Political, ethnic, cultural, and socio-economic are the main reasons for the inequity. Politically, disparities may arise from unequal distribution of resources and varying levels of health-care funding across regions. Most of the time, minority groups may face barriers

because of language differences, discrimination, and cultural insensitivity in health-care systems. People from low-income group are highly exposed to health risks and struggle with limited access to quality health-care and insufficient insurance coverage.

- **Infrastructure Issue:** Lack of transport, electricity, and internet facilities are the reasons for poor health-care systems in rural areas. Sometimes the natural condition in some areas is also harsh which creates a barrier to improving the health-care system for these areas. Environmental pollution is also a major reason for creating a proper health-care system.

- **Lack of training for the health-care professionals:** Sometimes there is no provision for training for health-care professionals due to lack of financial resources. The use of advanced medical tools and devices in health-care centres is only possible through medical experts.

2.11 Summary

eHealth stands at the forefront of a health-care revolution, offering a holistic and patient-centric approach to medical care. As technology continues to advance, the integration of eHealth solutions will become increasingly vital in shaping the future of health-care delivery. By addressing challenges and fostering collaboration between health-care providers, policymakers, and technology developers, the full potential of eHealth can be harnessed to create a health-care landscape that is efficient, accessible, and personalized for every individual. In embracing eHealth, a journey towards a healthier and more connected world may be initiated.

In the next Chapter 3, the a real-time ICT based application has been discussed elaborately.

Remote monitoring technologies hold significant promise for improving patient outcomes, particularly for those with chronic conditions.

— Eric J. Topol, M.D., Cardiologist, Scientist, and author of The Patient Will See You Now

3

A real-time application based on sensor, mobile and cloud technology

Contents

3.1 Prelude

Delivery of primary health-care services to the doorstep of rural people in developing countries like India is a challenging task. It requires a huge skilled workforce along with proper medical equipment for medical diagnosis and monitoring. Primary health-care centres (PHCs) are set up in rural and remote areas to provide health-care services to the majority of the rural people. But these centres suffer from a

lack of resources, including doctors [81], health workers and all types of clinical facilities. This scarcity of resources can turn out to be disastrous for the rural people. For example, during the COVID-19 pandemic, it was observed that no steps could be taken in the vast rural area of the country where hundreds of thousands of migrant workers from the cities returned. It became difficult for the administration to track these people and to deliver basic health-care to the villagers living in these areas. The village health workers, such as ASHA (Accredited Social Health Activist) and ANM (Auxiliary Nurse Midwife) had been engaged to handle this situation. However, in the absence of proper guidance, even preliminary care and advice became almost impossible. Moreover, after the spread of Corona pandemic, telemedicine has been considered to be an effective solution as hospitals remain at a risk of spreading COVID-19 among co-patients and visitors and their family members. In countries which have limited medical resources, doctors should not be infected during this pandemic, and therefore telemedicine or mobile apps can be effectively used as pointed out by some telemedicine workers.

Medical web portals [114] and telemedicine apps have been developed to help health assistants working in rural centres to provide information about treatment and follow-up actions. However, such portals are focused on only dissemination of relevant information. They are unable to provide assistance for vital measurements and to capture required information for proper treatment. When doctors are not available at the PHCs, rural health assistants provide assistance for the treatment and in case of emergency, they contact the urban doctors using phone calls. This is certainly not an effective way for health-care delivery.

To overcome this issue a highly modular easily re-configurable touchscreen-based application KiORH (**Ki**osk **O**perated **R**ural **H**ealthcare) has been developed based on sensor, mobile and cloud technologies with a user-friendly interface for use in rural areas where none or minimum health facilities are available. A kiosk can be set up in the villages of India where health assistants, with the help of the application, measure the patient's vitals, collect symptoms by using a knowledge base and perform clinical examinations. The collected data will be stored in a cloud

database server. Sitting at the urban location doctors can check and examine the patients' clinical data, and prescribe medicines and medical tests. The following subsection discusses the current state of the art and provides details about KiORH.

3.2 Present state of the art

In the current scenario, kiosk-based health-care startup is transforming health-care in rural areas. In this section, a few well-known applications for rural health and kiosk-based health-care are introduced [18].

- Gramin Healthcare is providing affordable primary health-care and specialist care in the rural and poorest regions of India. This *Gramin Healthcare* kiosk provides diagnosis, subsidised doctor consultation by live online video/ audio chat and medicines through their health-care platform [63].

- iKure Health Monitoring kiosks were set up in rural areas of West Bengal, India where health-care infrastructure is negligible. These kiosks have been set up by iKure Techsoft, in partnership with Aegle Angels Foundation. The kiosks are equipped with WHIMS—the wireless health incident monitoring system which can capture data from a maximum of 16 medical instruments or 16 patients at the same time. The WHIMS reads measurements from these instruments, and wirelessly transfers and stores them in a data server as the patient's medical history [21]. Based on the test results, the Registered Medical Practitioner (RMP) can prescribe medicines.

- e-Mitra is a kiosk-based health-care delivery solution for rural areas, launched by the government of Rajasthan, India. Through this application, an expert doctor across the globe attends the patients in rural areas. The e-Mitra service allows a patient to write a health query, along with pictures, a lab report etc. and post it to doctors across the world and also get answers online or over the phone from the experts [19].

- HealthSpot kiosks provide real-time access to health-care professionals using advanced medical devices and diagnostic tools for remote consultation. Patients could visit a HealthSpot kiosk in a pharmacy or other public location in US, where they could connect with a doctor via video conference, receive a diagnosis, and even get a prescription. But currently it is not operational.

- The SoloHealth Station is a self-service health kiosk that offers free health screenings for vision, blood pressure, weight, and body mass index (BMI). The Kiosk are located in retail locations like pharmacies and grocery stores of United States [10].

- Higi Stations are health kiosks found in various retail locations that offer free health screenings, including blood pressure, pulse, weight, and BMI by integrating with various health and fitness apps, enabling users to create a comprehensive health profile and receive personalized health recommendations [11].

- PharmaSmart kiosks are designed to provide accurate blood pressure readings and health information in pharmacies in United States and Canada [12]

- MedAvail MedCenter kiosks are automated pharmacy kiosks that allow patients to fill prescriptions, consult with pharmacists via video conferencing, and receive health-care services in hospitals, clinics, and retail stores in the US or in Canada [13].

- M-TIBA kiosks of Pharma Access provide health-care services, including health-care payments, health monitoring, and telemedicine consultations in Kenya [14].

- Healthpoint kiosks in New Zealand provide health information and services, including health check-ups and telehealth consultations [20].

- MedVault kiosks in South Africa provide health screenings and telehealth services, helping to bridge the gap in health-care access in remote and underserved areas.

- SmartHealth kiosks in Australia offer a range of health screenings, including blood pressure, BMI, and glucose levels.

There are many other mobile-based health-care applications available in the market. Some well-known and popular applications are

- Dr. Lal PathLabs is an Indian service provider offering diagnostic and related health-care tests on blood, urine, and other human body tissues. [15]

- Practo offers consultations with India's top doctors via video consultation, provides digital prescriptions, allows a patient to order medicines, and lets you book appointments of the doctors and lab tests. [1],

- Credihealth is a leading health-care platform in India. With a network of doctors and hospitals, a patient can find the right care at low price. Appointments can be booked to get treatment. [2],

- Tweet2Health is a popular social health network for trusted medical practitioners and information,

- MedicExpress focuses on making health insurance benefits hassle-free, accessible and affordable to every citizen in Malasia.

- MD-Calc is a free online medical reference for health-care professionals that provides point-of-care clinical decision-support tool [6],

Curofy [3], Healthonphone, Symptomate Symptom Checker [4], Prognosis: Your Diagnosis [5] work fine in places where enough network bandwidth is available. However, these applications are unable to offer a coherent procedure to enable health-care professionals or doctors to treat patients remotely by gathering precise information about the symptoms using a previously gathered knowledge base.

3.3 KiORH - Kiosk based Remote Health-care

The KiORH application has been developed for the rural kiosk centre where resources like Internet connectivity and electricity are big constraints. The application has two main sections. One is for the kiosk side, from where the kiosk operations are carried out. There are five modules in this section for performing five different tasks. Another section is at the doctor's side, from where the doctor can log in, check vitals of the patient and make prescriptions.

To operate the kiosks, health assistants (rural girls) can be trained to use information and communication technologies (ICT) and to collect necessary data from the patients. Android-based mobile phones and tablets are used at the kiosk side. On the other side of the application, doctors can view the complaint of the patient, along with other details like vitals, history and even old prescriptions and relevant reports. Based on these data and through remote interactions over the Internet, a doctor can put forward his/her advice for the patient and prescribe medicines. A high-level overview of the kiosk-based operation is shown in Figure 3.1.

Figure 3.1: KiORH: Kiosk Operated Rural Health-care delivery

3.3.1 Overview of the KiORH application

When a patient visits a kiosk, the health assistant measures clinical parameters, like blood pressure, SpO2, and pulse rate along with some other vitals of the patient using eHealth sensors and collects demographic data of the patient. The data are sent to the application running on the Android tablet/mobile phone through Bluetooth. In the next step, the health assistant asks various questions depending on the symptoms of the patient. The questions are displayed on the app screen from a knowledge base and the health assistant selects the appropriate answer in the provided space on the application screen. After gathering the present symptoms of the patient, the health assistant queries about family history, and patient's habits, gets images of old prescriptions and other medical documents and inputs these data/documents to the app. Next, the application guides the health assistant to make a clinical examination of the patient. The application analyzes the details given by the patient and based on this analysis, it displays a list of clinical examinations which are required to be performed for the particular patient. The health assistant carries out the examination, puts the results in the provided space in the app and selects a doctor from a pool of currently available doctors. The data is then uploaded to the cloud server for further use by the doctor.

The work-flow of the kiosk-side application is shown in Figure 3.2

The Kiosk side application is developed using HTML5, CSS3 and JQuery3.x wrapping with Cordova [34] (formerly PhoneGap) into a native container which can access the device functions on several platforms. To run, develop, and test the KiORH app using Cordova, the primary focus has been on Android devices such as Android tablets and smartphones are used, because these are relatively inexpensive and easily available in the market. For cloud storage, VPS hosting with 2GB RAM, 2vCPU and 1000GB primary bandwidth have been used.

3.3.2 Data model used

Initially, a data model was built for primary health-care delivery was built by us. Data modelling for the health-care domain is difficult because the health-care

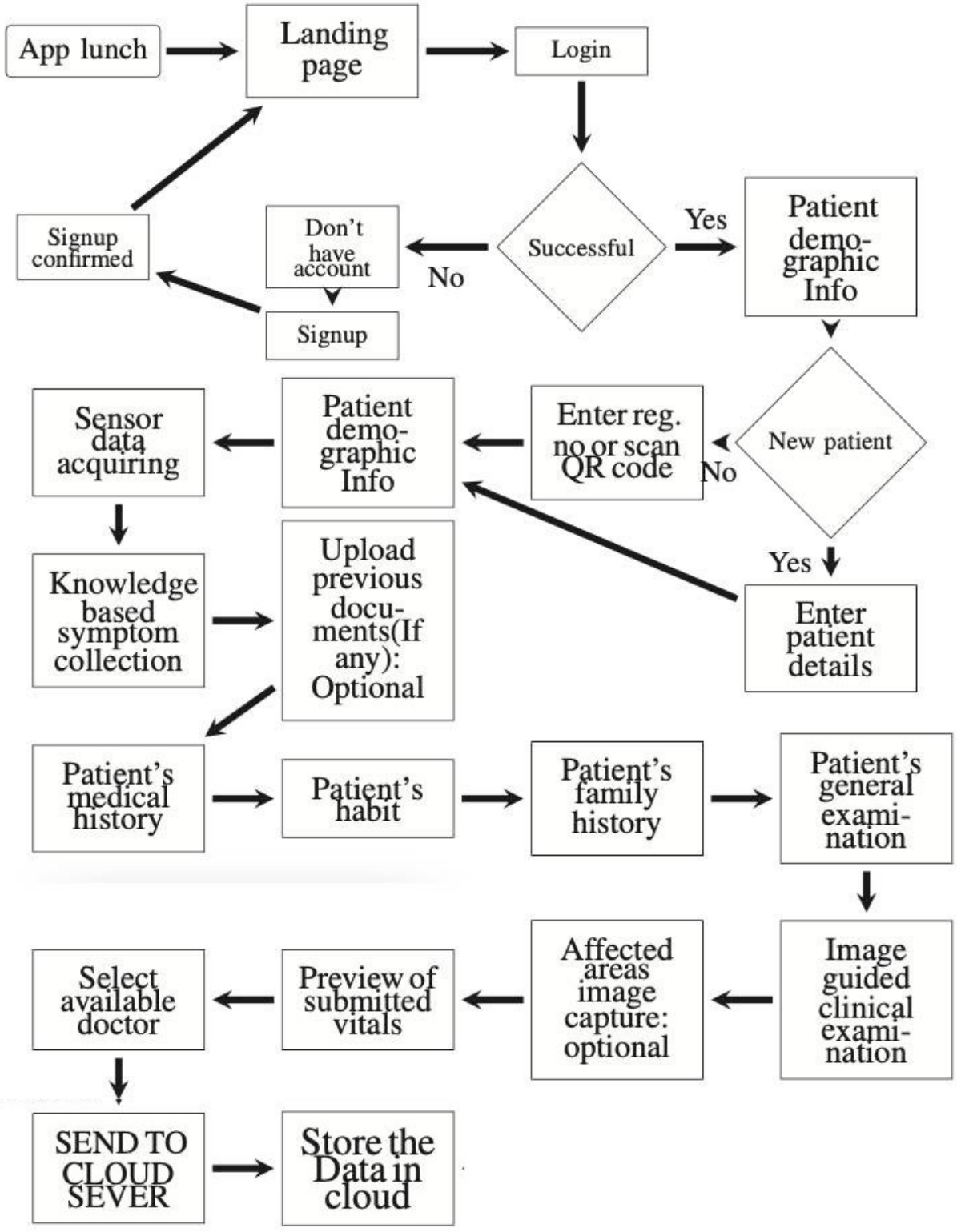

Figure 3.2: KiORH application workflow

domain is both broad and deep. The number of data concepts and ways in which they can relate to one another is shown in the Figure 3.3. There are two different participants, such as patients and physicians. Each participant has a slightly different view of the care processes they are involved in.

After incorporating the participants, problems arise due to the complex nature of clinical knowledge and how very few processes are perfectly repeatable across time and patient populations. As the main objective is to model primary health-care, a simple data model [103] for health-care management is used. Figure 3.3 shows the data-model followed in this thesis.

3.3.3 Features of KiORH

Following are the notable features of KiORH that make it different from other similar applications and also suitable for the developing countries where scarcity of resources is a challenge.

- **Role-based login:** Role based login is the first module of the application. This module authenticates valid users to use the application. The module connects with the cloud server via REST API. It sends the username and password of the user (health assistant/patient/doctor) to the server and fetches the user details as a response. If the application gets a positive valid response from the server, it stores the fetched user (e.g. health assistant) information in the local memory for further use. The role-based login allows the users to access only a part of the system and prohibits them from accessing other parts which they are not authorized to access.

- **Sensor data acquisition:** Sensor data gathering and real-time flow from the kiosk to the doctor's end is another important activity of the kiosk application. A module acquires sensor data of the patient who is currently under checkup using e-health sensor devices through Bluetooth connection and sends the data continuously or the last seen record to the cloud server for further use by the doctor. Presently, the Android application installed on the mobile device

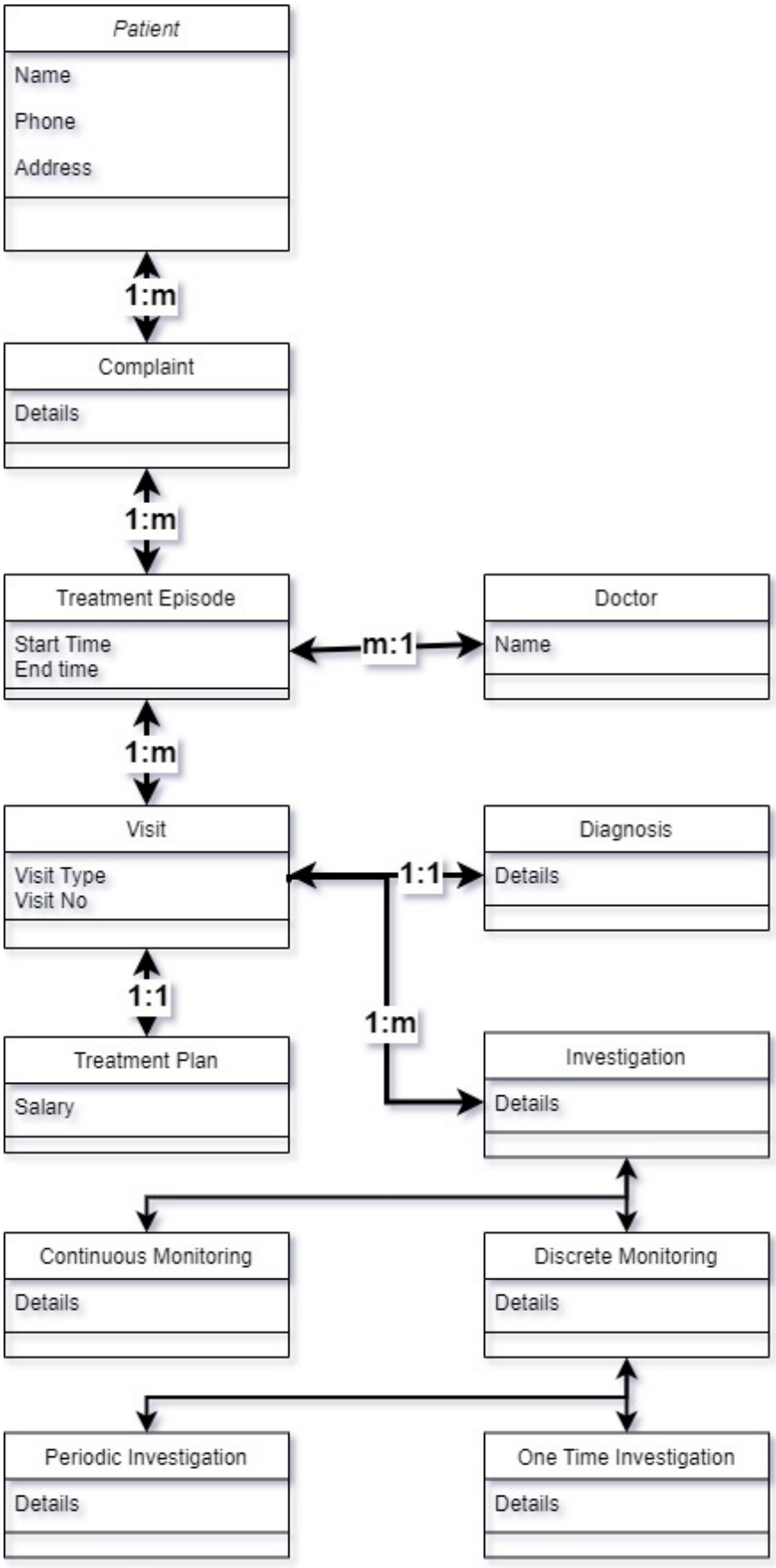

Figure 3.3: Class diagram of the Data Model

communicates with the sensors via USB serial communication. Sensors are attached with an Arduino board which sends the sensed data through the USB port. The Arduino board is equipped with the eHealth shield. The set of sensors attached to the shield includes a body temperature sensor, SPO2 sensor, airflow sensor, and blood pressure sensor that comes bundled with the eHealth sensor kit as shown in Figure 3.4. Figure 3.5 displays the sensor data collected from a patient.

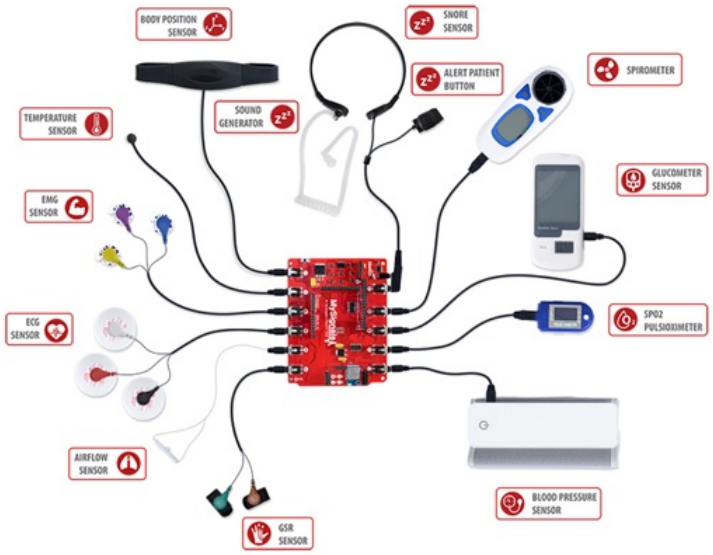

Figure 3.4: Health Sensors

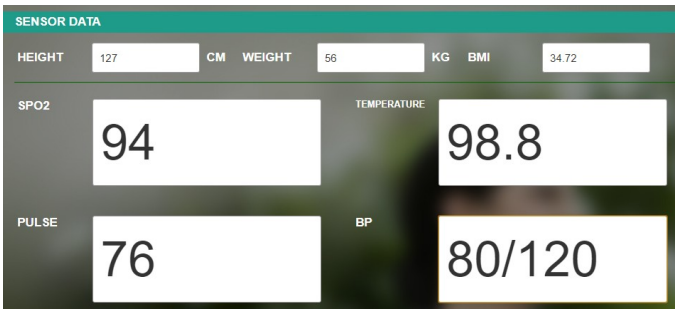

Figure 3.5: Sensor Data Screen

- **Symptom collection:** The kiosk-side android application uses a knowledge base for collecting symptoms and other details of the patient. In the first step, multiple main symptoms are listed. The health assistant can select one

or more main symptoms by tapping the app screen based on the patient's complaint. After the selection of symptoms, sub-symptoms are shown on the app screen. Next, after the selection of sub-symptoms, the currently selected symptoms and sub-symptoms-related questions are displayed along with multiple possible answers.

An example of symptom collection for a patient with a headache using the knowledge base is given below. At first, from the list of symptoms, like *Pain, Swelling, Difficulty in breathing* etc, the health assistant selects *Pain*. Then the list of sub-symptoms of *Pain* will show up. The list includes the symptoms *Headache, Ear Pain, Teeth Pain, Throat Pain, Back Pain* etc. The health assistant selects 'Headache' as sub symptom. After this selection, the symptom-related questions like *duration, starting location, current location, intensity, nature* etc. will be displayed along with multiple answers to each question. Answers to all the related questions are to be entered by the health assistant. Figures 3.6, 3.7 show the screens displayed by this module.

A multidimensional hash table is used for storing the knowledge base. The hash table maintains key-value pairs for storing the details of all symptoms, sub-symptoms and related questions and answers. The idea of hashing is to distribute entries (key-value pairs) uniformly across an array. Each symptom is assigned with a key. There are also multiple keys for maintaining the sub-symptoms of the symptoms and similarly, there may be multiple keys for questions and a set of possible answers under sub-symptoms. Hash functions are used for fetching from the hash table, allowing keys (symptom keys) to be passed as arguments and symptom details to be returned as hash values. The structure follows the JSON format for storing the symptoms and their details. The grammar of the knowledge-base JSON document is shown below.

$$
\begin{aligned}
&Knowlegdebase =< Symptom, ClinicalExam > \\
&Symptom =< Category^{+} > \\
&Category =< id, title, Subcategory^{+} > \\
&Subcategory =< id, title, SymptomQuestion^{+} > \\
&SymptomQuestion =< id, title, type, answer^{+} > \\
&ClinicalExam =< ExamCategory^{+} > \\
&ExamCategory =< id, title, ClinicalQuestion^{+} > \\
&ClinicalQuestion =< id, title, answer^{+} > \\
&type = string \,|\, numeric \,|\, select \,|\, check \,|\, radio
\end{aligned}
\tag{3.1}
$$

- **Multilingual interface:** Since the application is intended for use by rural people to provide primary health-care facilities in rural areas, local language interfaces are required. Most of the rural people are not efficient in the English language and are free to use their native regional language. So, the facility to use the kiosk application in regional languages is provided There is a module with which the application provides multilingual support to the app and the health assistant can switch to any regional language. Currently, only "Bengali" support is set up and installed as shown in Figure 3.7.

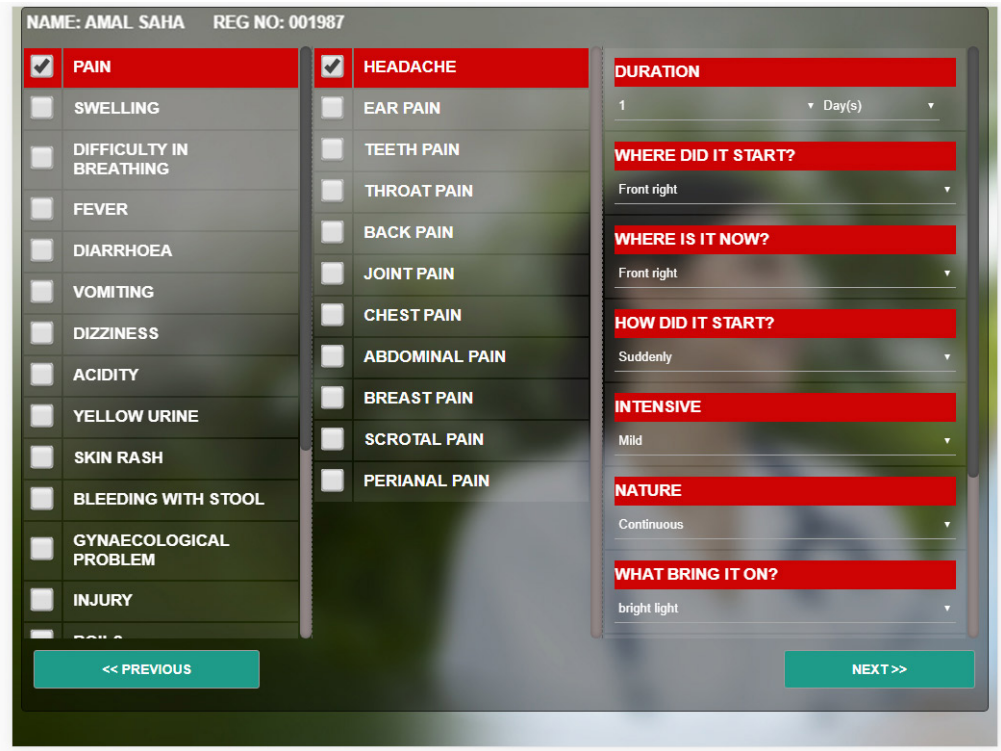

Figure 3.6: Symptom gathering screen (English)

Figure 3.7: Symptom gathering screen (Bengali)

For the entire application, a JSON table is maintained to represent multilingual texts. Within the table, a collection of different languages supported by the application is maintained. Each language has a dictionary that maps a fixed set of keys to values corresponding to that language. As the set of keys is fixed for all languages, these dictionaries can be queried to retrieve the phrases corresponding to a specific language by providing the key and language title. A method takes the 'language' and the 'key' as arguments and returns the text phrase from the JSON table. The grammar of the language as a JSON document is shown below.

$$\begin{aligned} table &=< language^{+} > \\ language &=< id, title, kv^{+} > \\ kv &=< key, value > \end{aligned} \tag{3.2}$$

- **SMS-based transmission:** Since, technological advancements happen slowly in rural and remote areas, ICT-based health-care services face many challenges.

One of the major constraints in rural areas is the non-availability of fast and reliable Internet connection. Hence, the idea of transmitting the patient's vitals to the cloud through SMS is used when the Internet connection is poor or unavailable. A separate module is used for this purpose. After the collection of all patient details, the application tries to send the data over the Internet. If there is no Internet connection, the application accumulates important patient's vital data (symptoms) in the form of JSON string, compresses the string and sends it via SMS to an SMS server. The SMS server sends the compressed data to the cloud server. Another application running on the cloud decompresses the data and changes it to the original JSON format [82]. The detailed discussion on this service is provided in Chapter 4 of the thesis.

- **Handling pandemic situation:** KiORH is a cloud-based modular application. A new knowledge base can be added easily anytime and the application can be extended to handle the pandemic situation. During the pandemic, the knowledge base was enriched by adding Coronavirus (COVID-19) assessment scan-related questions and options, based on guidelines from WHO [16] and the Ministry of Health and Family Welfare (MHFW), Government of India [79]. This assessment can help the health assistants to identify the COVID-affected patients and ask them for quarantine. Currently, only English language support is set up as shown in Figures 3.8 and 3.9.

3.4 Connecting doctors

On the other side of the health kiosk, there is a pool of doctors having access to the other part of the application. When a health assistant connects a doctor from the pool, and a complaint is created, the selected doctor (selected at the kiosk side) is notified. The doctor can view the patient's complaints and all related information. Figure 3.10 shows a patient‘s complaint and its corresponding prescription as it appears on the web interface. Doctors can check the complaints along with the records of that patient which are uploaded to the cloud by the health

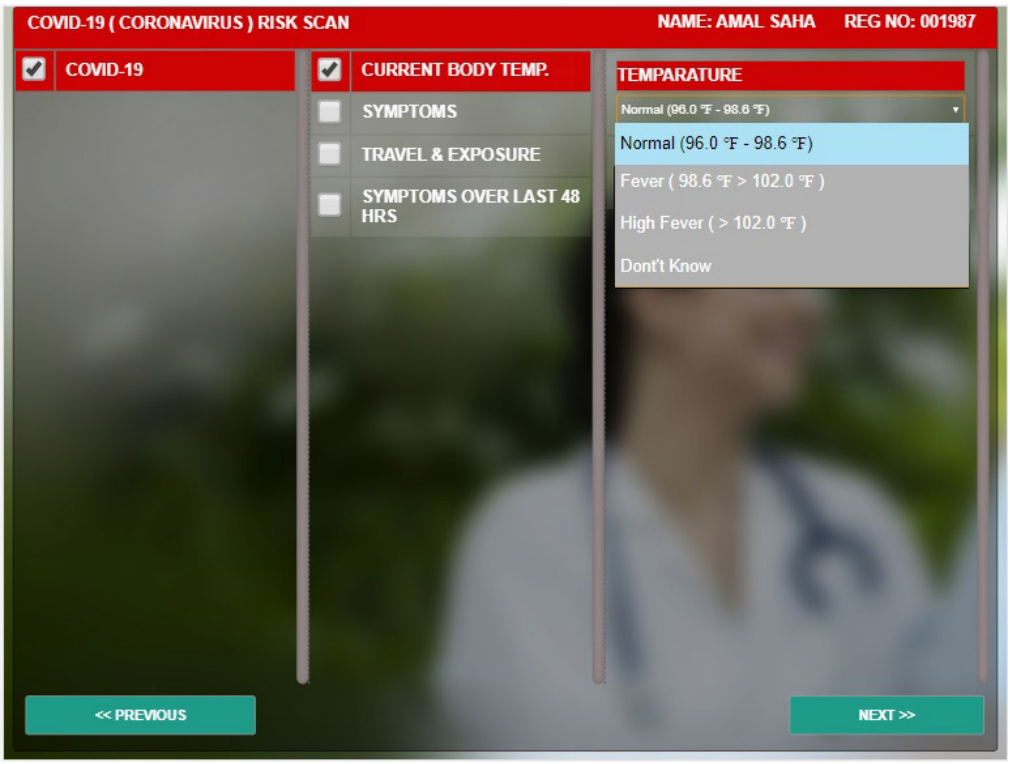

Figure 3.8: COVID-19 assessment screen 1

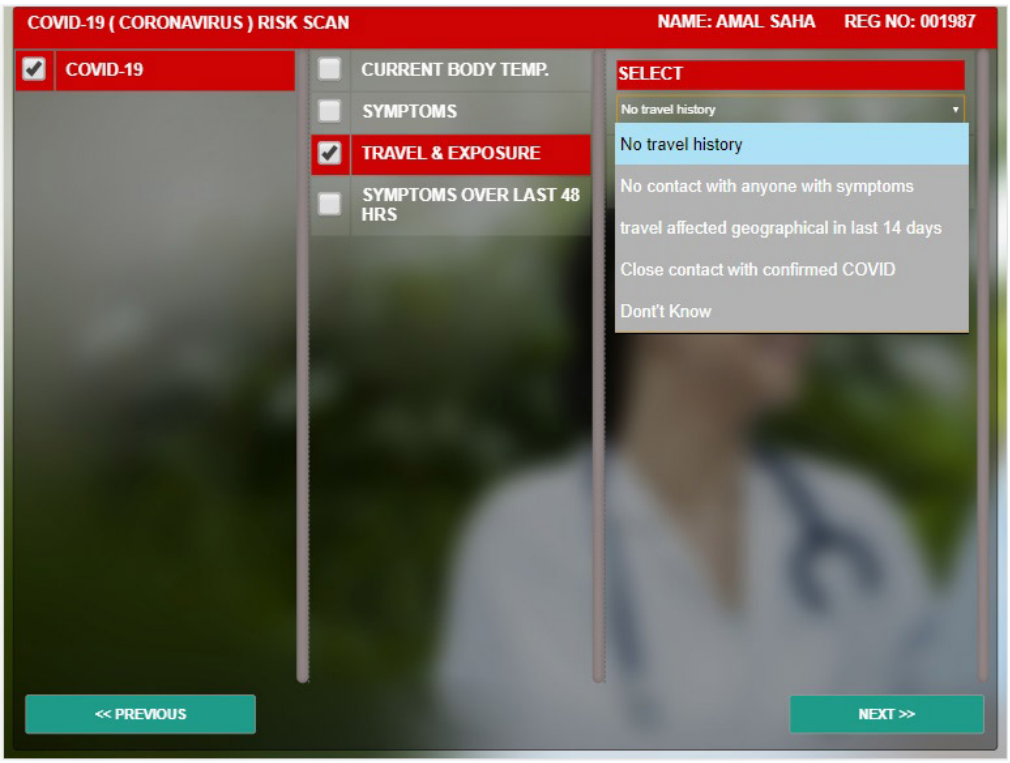

Figure 3.9: COVID-19 assessment screen 2

assistant. Medications, investigations and other advice can be prescribed. The web interface also provides a WebRTC-based audio video chat facility, which can be used for communication between doctors, patients and health assistants. For the convenience of patients and the rural health assistants, questions associated are asked in local languages. However, on the doctors' screen, these questions are represented in English language.

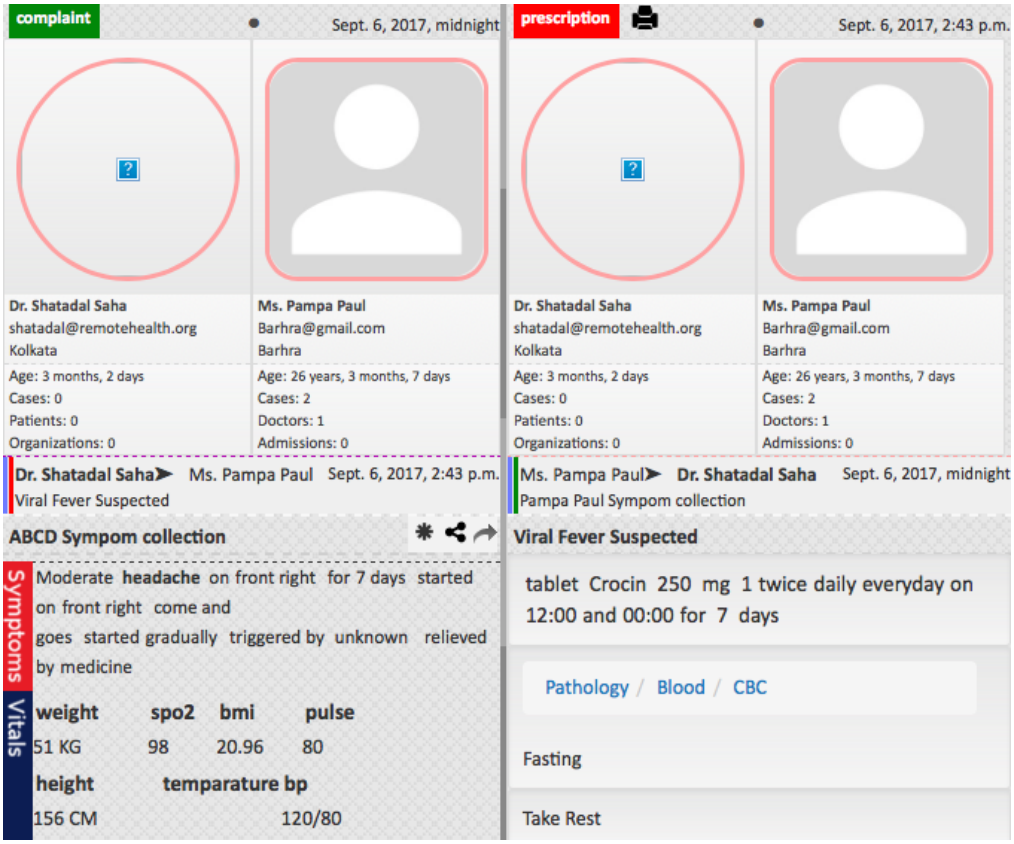

Figure 3.10: Complaint screen

Key-value pairs in complaints questionnaires could be represented in a tabular form to the doctor. However, for convenience, a string is generated out of the key-value pairs as shown in Figure 3.11. Similarly, medicines and advice are entered in a generalized structure which is represented as a string in the prescription. The home page of the doctor-side interface shows an overview of the recent admissions, complaints, prescriptions, patients and appointments. After selecting a patient, the complaints and prescriptions of that patient can be viewed by a registered doctor at any time anywhere.

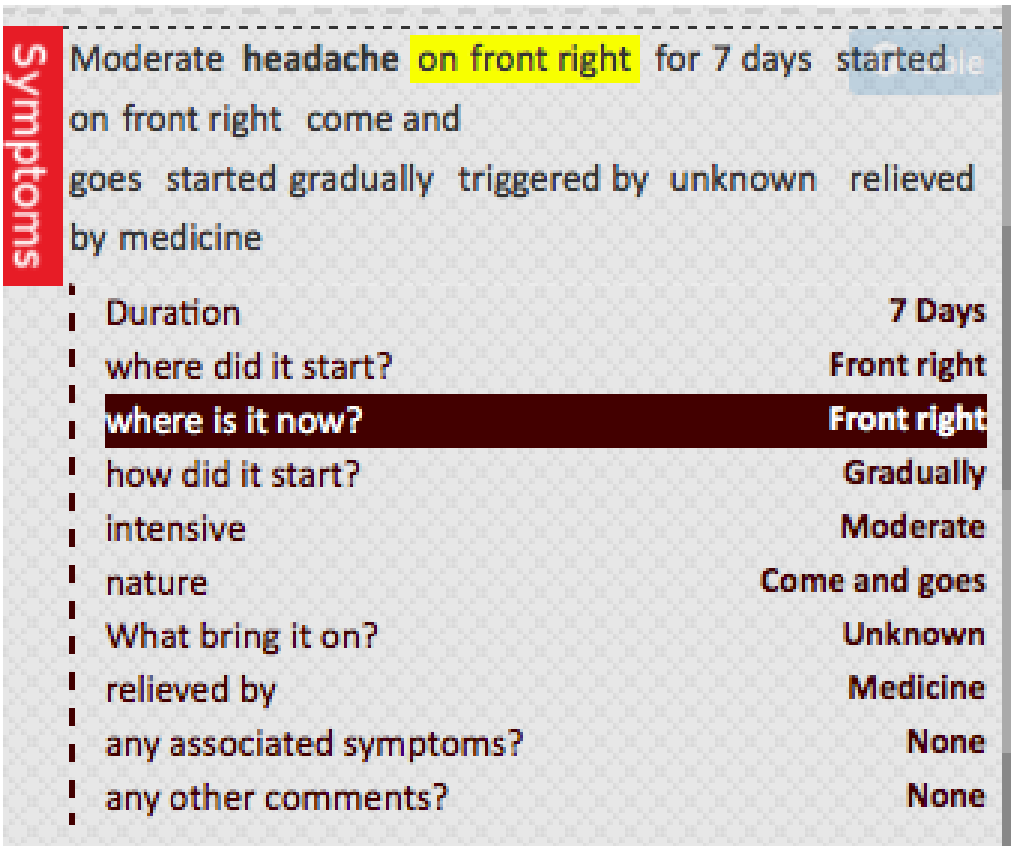

Figure 3.11: Details of Complaint

3.4.1 Prescription generation

Prescriptions at the doctor's end are composed in English. However, the medications and instructions for medications are shown using a bilingual interface to the patient. Doctors can also prescribe medical tests through the web interfaces as shown in Figures 3.12 and 3.13. Medicines are filled up by the doctor using a web interface. Some of these fields (such as type, name, dose, unit, termination, and count) are mandatory. For the remaining fields, a value may not be provided if that field is not applicable. If the mode of medication is clock-based, then a set of times is entered in the *clocks* field. If the mode of medication is event-based, a *when* field and a *context* field are specified (e.g. "after" (when) and "dinner" (context)).

However, the prescription needs to be composed in languages which are understood by the patients as shown in Figures 3.15, and 3.16. As there may be large variations in the grammar of two different languages (e.g. in the case of

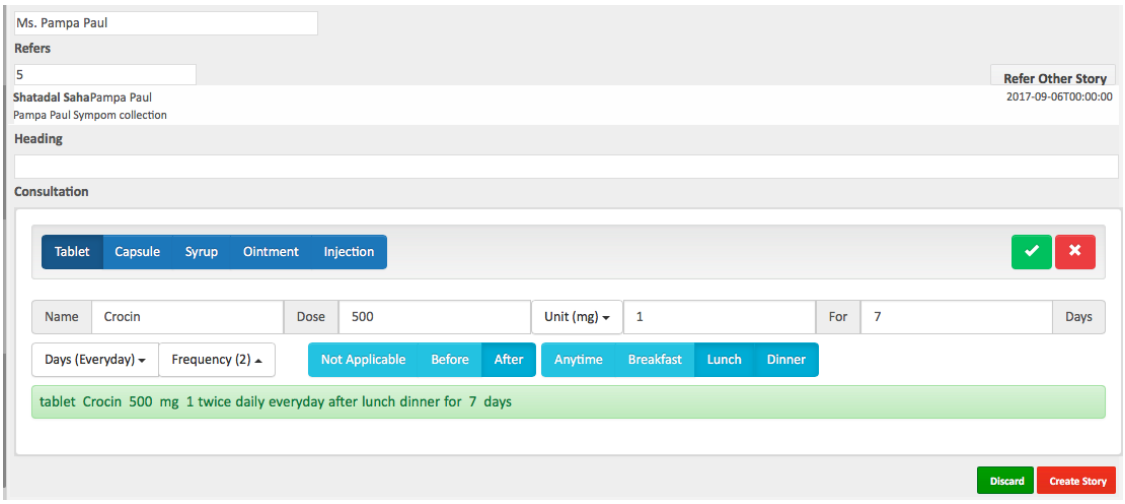

Figure 3.12: Prescription Composer: prescribing medicine

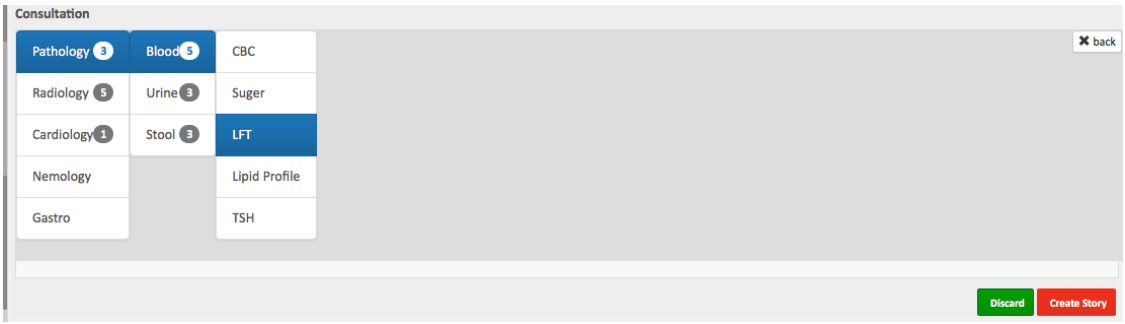

Figure 3.13: Prescription Composer: adding investigation

English and Bengali), the composition of a string from the data entered in the prescriptions for every language should be done differently. A meaningful translation can be achieved by substituting the placeholders with values of the keys which are shown in Figure 3.14.

At the kiosk side, the advice and the prescribed medications can be printed in user-accessible language. Every prescription has a unique QR code printed on its top right corner which can be used for referencing in future.

3.4.2 Interface between doctor, health assistant and patient

Once a prescription is created by the doctor, health assistants can view and print the prescription. Before printing, health assistants may also discuss the prescription or request for a change of medicines, which may lead to the regeneration of the prescription. For additional discussions, doctors, patients and health assistants can

```
type: '', // tablet | capsule | ointment
name: '', // name of the medicine
dose: 0,  // dose
unit: '', // unit of dose
termination: 0, // for how many days
count: 1, // how many
interval: {
    frequency: '', // numeric
    days: 0, // days interval
    mode: '',    // clock or event
    clocks: [],
    event: {
        context: [], // Lunch | Dinner
        when: '' // before | after | N/A
    },
    custom: ''
},
note: '' // some extra notes if required
```

Figure 3.14: Medicine Definition

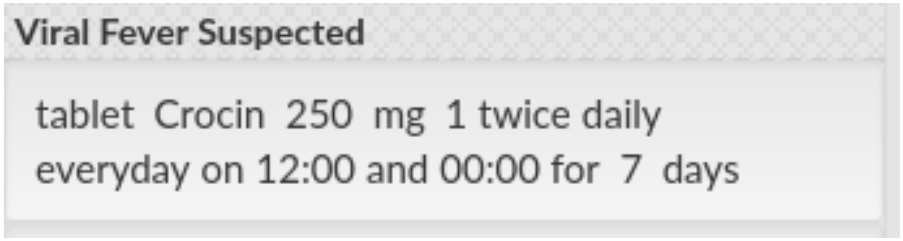

Figure 3.15: Translation of prescription in English

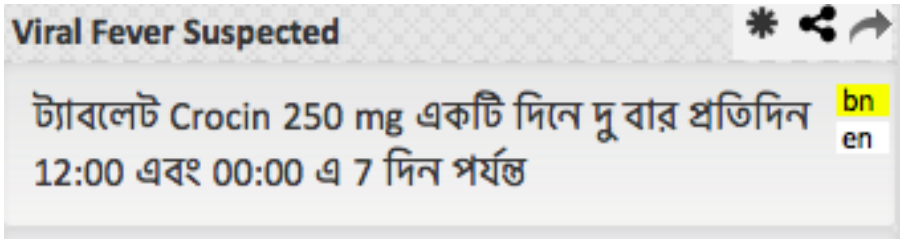

Figure 3.16: Translation of prescription in Bengali

interact over live chat provided by the application. The peer-to-peer audio and video conferencing via WebRTC is implemented this purpose .

3.5 Assessment of usefulness of KiORH

After two years of operation of the kiosk-based health-care delivery system, a survey has been conducted among the rural population to asses the acceptance of the KiORH application. The detailed discussion on this survey is made in the Appendix A

3.6 Summary

The integration of eHealth and mHealth through ICT is revolutionizing healthcare by improving accessibility and efficiency, particularly in rural areas. While eHealth utilizes Electronic Health Records (EHRs) and telemedicine, mHealth relies on smartphones and wearable devices for remote consultations and health monitoring. ICT based KiORH (Kiosk Operated Rural Healthcare) application was developed to address healthcare challenges in resource-limited rural settings, enabling health assistants to collect patient data using sensors and Android devices, and allowing urban doctors to provide remote care via cloud-based systems. Health assistants, trained to use ICT, gather vitals like blood pressure and SpO2, collect symptoms from a knowledge base, and upload data to the cloud for doctors' analysis. The application ensures real-time sensor data transmission via Bluetooth and supports role-based logins for secure access. The application also features a multilingual interface offering Bengali as well, utilizing a JSON-based system for storing and displaying symptoms, sub-symptoms, and related questions. KiORH is a scalable solution for improving rural healthcare delivery in developing countries like India.

Next chapter focuses on communication of health data in a remote area where internet connectivity is a problem. In Chapter 4, approaches for real-time video transmission using mobile ad-hoc network and transmission of health data using SMS, will be discussed elaborately.

The future of health-care is not about more technology, it's about better technology.

— Dr. Atul Gawande, Author and Surgeon

4

Data transmission for remote health-care services

Contents

4.1 Prelude

In the rapidly evolving health-care landscape, the convergence of technology and data has become a pivotal force driving innovation and efficiency. Among the transformative advancements, integrating Big Data in health-care has emerged as a cornerstone, revolutionizing how information is collected, analyzed, and transmitted. Health-care-related video transmission is a burgeoning field poised to transform how medical data is shared, analyzed, and leveraged for improved patient care. This technology transcends traditional video communication, encompassing the secure and efficient transfer of massive datasets, including medical images, genomic sequences, and real-time sensor readings.

In disaster management, the ability to communicate and coordinate effectively during emergencies is a difficult process. In the disaster scenarios, traditional communication infrastructure, such as the internet and cellular networks, often becomes unreliable or entirely unavailable. This disruption causes significant challenges for rescue operation and attempt of medical personnel to respond swiftly. In this context, video capture and streaming play an important role. However, in absence of proper communication infrastructure for streaming applications, Mobile ad-hoc Network (MANET) is considered to be the only solution. The routing protocol, video frame size and quality have a vital impact on the QoS of MANET. To address these challenges, innovative techniques are required to ensure that critical data can still be transmitted and that real-time situational awareness can be maintained. For rescue purpose, for speedy and accurate medical diagnoses and remote consultations, health-care-related video transmission over MANET is important.

This chapter aims at efficient Big Data video transmission of health-care data, even when Internet connectivity is not available. For video transmission, a cross-layered fragmentation approach is used to transmit high-resolution real-time

multimedia data with minimal delay, high throughput and high fps despite limited bandwidth. Here image frames or video frames are segmented into smaller chunks and sent to the network as a payload of a network packet. This ensures that the data lost in the network is less compared to non-segmented frames. Infrastructure-less Mobile ad-hoc Networks (MANET) are used as a network transmission media to test medical video transmission without internal connectivity or unreliable networks.

Along with the video transmission, this chapter also describes an approach to connect rural patients with urban-located doctors using SMS. The chapter provides evidences that even when Internet connectivity is not available in the villages, it is still possible to actively use e-health-care services for treatment of the rural patients. For transmission of health data between the rural patients sitting at rural health kiosks and Doctors consulting from their urban chambers communication is established through SMS by encrypting and compressing the data.

By combining MANET-based video transmission and SMS-based data transmission, this framework ensures that channels remain open during disasters, enabling better coordination, proper data transfer, and more efficient communication. The chapter will elaborate on the technical capabilities of these techniques, which can be used to improve disaster response operations through real-time communication and data-sharing in adverse conditions.

In the first part of this chapter, the video transmission using mobile ad-hoc network is discussed and in the second part, an approach to transfer health data from one end to another using SMS is discussed. In both cases, data is transferred without Internet connectivity or a structured network interface.

4.2 Mobile ad-hoc network

Mobile ad-hoc Networks (MANETs) represent a dynamic and self-configuring category of wireless networks where computational nodes, equipped with wireless communication technology, collaboratively establish a communication infrastructure without relying on any pre-existing centralized authority. Unlike traditional networks that rely on fixed towers and cables, MANETs are self-organizing webs of devices,

dynamically connecting and sharing resources without any central control. MANETs are characterized by their ability to operate in scenarios where fixed infrastructure is impractical or non-existent, making them particularly well-suited for military operations, disaster response, and ad-hoc collaboration in temporary environments. Nodes in a MANET can function as both end systems and routers, facilitating multi-hop communication to transmit data across the network. The absence of a fixed infrastructure introduces several challenges, including routing complexities, limited bandwidth, and the need for efficient protocols to adapt to the dynamic topology changes that occur as nodes move.

4.3 Data transmission over MANET

In recent days, the health-care domain has witnessed explosive growth in the use of multimedia applications, such as audio and video streaming with mobile devices. In certain situations, for example, in disaster scenarios or battlefields, applications involving multimedia data streaming over mobile ad-hoc networks (MANET) can be developed and deployed for use by the disaster recovery team or by the army. Video streaming over MANETs can provide a wide range of benefits, including improved accessibility, increased flexibility, lower costs, faster deployment, and increased privacy. These benefits can make video streaming a viable option in situations where traditional network infrastructure may not be available. MANETs can be quickly deployed in emergencies or other scenarios where time is critical. As the network is decentralized and does not rely on centralized servers, users can stream video content without the need for a third-party provider, which can increase privacy and security as well. However, mobile ad-hoc networks are resource-constrained and bandwidth-constrained. Therefore, transferring multimedia data through a multi-hop ad-hoc network is a big challenge for low network bandwidth and unstable mobile devices. Moreover, real-time streaming of videos over a MANET poses more challenges due to the uncertainty of the network path. For example, if the streaming video consists of 30 frames per second and the captured video resolution is 800 pixels X 600 pixels, then the 30 still images with the resolution of 800 pixels X

600 pixels will be transferred in every second. This will require a huge bandwidth and a stable network path to transfer the data from source to destination via some intermediate nodes. In real-life scenarios, each image or video frame is large and cannot be transferred in one payload of a network packet. The maximum size of an IP packet payload is 65,535 bytes or 64 kilobytes. Whereas the maximum size of an Ethernet packet or "frame" is only 1,500 bytes or 1.5 kilobytes. So the image frames or the video frames are divided into smaller chunks and sent to the network as payloads of several network packets. At the destination end, the payloads of the packets are combined to get back the original data. During transmission, network packets may be dropped or corrupted as wireless links are repeatedly broken and re-established. For this reason, the entire image frame cannot be assembled to get back the exact original frame, because of the intermediate packet losses.In the case of huge number of packet losses, the entire frame needs to be discarded. If the source node or intermediate nodes change their locations, the network becomes unstable and needs time to re-establish the network path. During the network reestablishment time, some packets may be lost leading to further errors in the transmitted data. Compared to textual data and sensor data, multimedia data, such as raw videos or raw captured images are large. Therefore, it is also required to compress the data before sending it. However, the mobile ad-hoc network, being an infrastructure-less network, sometimes becomes noisy and the noise needs to be filtered out. In such cases, often the corrupted frames need to be discarded, and it becomes impossible to reconstruct the frames from the compressed frames. Thus, for all the above reasons, the FPS (Frame per second) rate drops drastically when the hop count increases. To overcome this problem, a cross-layered fragmentation scheme is used for transferring high-resolution video frames over MANET. The fragmentation is done in the application layer and only the relevant parts are transmitted, thereby requiring less network bandwidth. Specific advantages of the proposed approach include reduction in packet loss, improving video resolution, and minimizing buffering time. The scheme can lead to faster data transfer and reduced energy consumption for mobile devices. This scheme

proves to be effective for video streaming over MANET with reduced time and space complexity. The scheme has been implemented for video transmission over an ad-hoc network based on BATMAN-adv protocol.

4.4 Recent advancement in the development of video transmission

Real-time video streaming is an important aspect of communication technology. Initially, the videos were captured and transmitted in analogue form. Real-time video transport requires minimal loss, reduce delay and adequate bandwidth which are not always available in ad-hoc networks. Wireless ad-hoc networks are prone to perennial link failure which can lead to video frame loss. To successfully transfer high-quality video through the ad-hoc network, some research works have been carried out in the last two decades. Most of the research works have been done in the protocol section. The researchers are trying to send the high-definition video by creating new network protocols or by modifying the existing protocols like AODV, DSR, DSDV etc. Some research has also been carried out for sending quality video by changing the format or compressing – decompressing or tweaking the video frames. The research works can be categorised as follows

- **Research works on protocol section -** Ivaylo Haratcherev et al. [52], presented the results of video-streaming over 802.11a environment in the presence of background traffic, generated by other stations sharing the same medium. They also showed that great improvements in the quality of video can be achieved by cross-layer signalling between the link layer and the video coder.

  H.C. Jang et al., in their paper [58], described their research on the hybrid design framework which consists of a MAC-centric cross-layered architecture to allow the MAC layer to retrieve video streaming packet information, a hybrid re-transmission deadline and retry limit to save unnecessary packet

waiting time and a single-video multi-level queue to prioritize packet delivery. They also carried out simulation in IEEE 802.11 and 802.11e environment.

Jian Liu et al., in their research article [75], considered three layers of the network protocol stack, application, data link and physical layers and proposed an Adaptive Cross-layer Mechanism for Real-time Video streaming (ACMRV), which includes both channel assignments and an adaptive Forward Error Correction mechanism. The proposed architecture has been evaluated via NS-2.

Hussein Muzahim Aziz et al. [29], proposed a new mechanism for streaming video over a wireless network. It prevents freezing frames in mobile devices by splitting the video frame into two sub-frames and combining them with another sub-frame from different sequence positions in the streaming video. If frames are lost or dropped, there is still a possibility that another half of the sub-frame will be received by the mobile device. The receiving sub-frames will be reconstructed to their original shape. Nor Khairiah Ibrahim et al. measured and studied the performance of video streaming over 802.11n, in 2.4GHz and 5GHz bandwith. The researchers measured the performance using Omnipeek and PRTG software [55].

Hui Zhou et al. developed a cellular-connected UAV testbed based on the Long Term Evolution (LTE) network with its uplink video transmission and downlink control and command (CC) transmission. They also designed novel algorithms for sending control signals and controlling UAV [123].

Yoshihisa Kondo et al. explained the concept, design, and prototype of Multi-Diversity MD-WLAN, and then demonstrated the low-latency live video streaming over MD-WLAN using the developed prototype system [71].

Thulani Phakathi et al. [91] conducted a simulation using OPNET 14.5 Modeler for comparing and analyzing four existing routing protocols in VANETs. The objective of this research was to identify efficient routing protocols that can deliver better quality of service (QoS) for video streaming applications

concerning the number of nodes used. The authors analyzed the results concerning throughput, delay and routing traffic received.

Another group of researchers proposed a short fountain-code-assisted UDP scheme for sustainable wireless high-definition video streaming. They integrated short fountain codes and UDP to form an efficient transmission control scheme. The authors claimed that using the proposed scheme, under the short-length constraints and the same redundancy level, fountain codes using degree distribution achieve lower decoding time and higher success rate compared to those using the classic solution of degree distribution [122].

- **Research works on video format and compression -** . The authors, Kazuo Takahata et al. [112] introduced an optimal control method to determine how to transmit the compressed video packets from server to client so that the requirement for network bandwidth is minimized and network input or output buffers can be used optimally. A client-server model for video streaming is proposed. The optimal sending data rate from the server is derived and the temporal buffer occupancy is calculated in this research work.

  The authors in [106] evaluate the performance of video transmission using H.264 video codec in two routing protocols [Neighbor-Aware Cluster-head (NAC) and Dynamic Source Routing (DSR)]. The authors used Reliable Multicast (RM) power model and minimized the delay to successfully transfer the video frames from source to destination. Gauss-Markov mobility model was used for the movement of Mobile Terminals (MT) in the experiment.

  In [60], the authors proposed to transfer video in UAV (Unmanned Aerial Vehicles) using FANET (Flying ad-hoc Network). Scalable Video Coding (SVC) is used for encoding the H.264 video. The authors checked the delay between the flying nodes. If the delay exceeds a certain threshold, the application reduces the quality of the video. If, however, the application measures a delay that is lower than the threshold, it will evaluate past delays and if it finds them low as well, it will try to increase the stream's quality.

Previous packets serve as "packet pairs" and their delay is used to estimate available link capacity. The technique works suitably because video stream packets are usually of the same size and sent back-to-back.

- **Research works on mobility -** A novel video source decision scheme for video delivery over VANETs, named COMIP (Cluster-based Overlay and Fast Handoff Mobile IP System), has been proposed by Jitendra Joshi et al. in [59]. Using Mobile IP without and with fast handoffs, the effects of mobility over the video transmission have been increased. The experiment was simulated in SUMO and NS-2.

- **Implementing AI on video streaming -**

  A video streaming scheme for Distributed Video coding (DVC) over a wireless ad-hoc network was proposed by authors Tian Bo et al. [33]. In this experiment, the neural network was designed and trained offline using different standard distributed video sequences to achieve generalization. The neutral network was exploited to predict the GOP (Group of Pictures) size of distributed video coding sequences, and the rate control method, which could be adaptive and could adjust the feedback time for blocks in the frame. The authors show that the quality of video in wireless mobile ad-hoc networks improved.

- **Research on ad-hoc network -** Multi-path video streaming is another approach to transfer the quality video from source to destination in the ad-hoc network. The author in [95] proposed a multi-path video streaming scheme that is relatively simple to implement and does not require modification of the H.264/Advanced Video Codec (AVC). The author has compared the proposed scheme with Video Redundancy Coding (VRC), which is a simplified method for Multiple Description Coding (MDC). They found that using 'redundant frames' results in superior video quality even if more packets are lost compared to VRC. This counter-intuitive result arises because some of those packets may carry redundant frames.

In another multi-path approach, the authors [65] introduced a technique for multi-path transport (MPT) with video compression and encoding. The video is split into multi-stream. These streams are transferred to particular routes using the routing protocol. At the receiver end, the video streams are placed in sequence and then the video stream is decoded and displayed.

Much of the above research work has been done using simulation. Only in a few cases, test-beds have been deployed for testing the video streaming. Testbeds have been developed using a maximum of 3 to 4 nodes in 1 to 2 hops. Researchers compared the distance vector routing protocol performance, namely AODV, DSDV and AOMDV with video streaming traffic on MANET with energy parameters, packet loss, throughput and jitter based on variations in network size [100].

On the other hand, in this chapter, a cross-layer fragmentation approach is propose and an application is developed which works on top of any protocols at the lower layers. The scheme has been tested on a test bed using BATMAN-adv.

4.5 Cross-layered video fragmentation and transmission

In this section, the design of a cross-layered video capture and streaming system, which improves the quality of streaming video over MANET is presented. A streaming video consists of multiple image frames. The image frames are transferred from the source to the destination one by one. While transferring to the destination, the frames are divided into small data packets. The size of the data packets depends on the defined MTU(Maximum Transmission Unit) or default MTU of the wireless interface.

The objective of the proposed scheme is to stream the video through the wireless ad-hoc channel. To trace and sniff the incoming data packets through the wireless channel, the monitor mode [93] was activated. Because of the unreliability of the ad-hoc channel, it is noticed through the monitor mode at the wireless network interface controller (WNIC) that, some of the data packets are missing at the

destination end. If the nodes (source node, destination node, intermediate nodes) are mobile, the chance of packet loss increases. Packet loss for a second can lead to low FPS and missing several video frames. To improve the video quality, the application-layer fragmentation of video frames is proposed. In this approach, the video is captured as image frames of size 1024 pixels X 768 pixels. In one second 28 to 32 image frames are captured. Each frame is then divided into sub-frames and is stored in a buffer. While sending the image frame, the current image sub-frame is compared with the previous subframe. If the change of pixels in the current sub-frame crosses a threshold limit, then the sub-frame is placed in the sending buffer. If the percentage of pixel changes is under the threshold limit, the sub-frame is not sent. This approach helps to stream large videos using a low bandwidth mobile ad-hoc network using a video streaming application. The next few subsections discuss the implementation of the proposed approach.

4.5.1 Fragmentation approach

The primary challenge in transmitting streaming video over a multi-hop wireless network is dealing with network congestion. The IEEE 802.11 standard, originally designed for static devices, struggles to handle fast-moving devices or changes in antenna orientation. As a result, network throughput fluctuates significantly, making it difficult to predict performance when devices are wirelessly connected and moving randomly within a given area. This variability in throughput often leads to sudden packet losses, which can severely impact video quality. Even the loss of a single network packet can result in the loss of several seconds of video playback [60].

To mitigate this problem, a cross-layered fragmentation approach for video transmission is proposed, aiming to improve both performance and quality of service (QoS). In this approach, video frames are divided into smaller segments based on the size of the network packets being transmitted. Normally, if a single packet is lost, the reconstruction of an entire video frame can fail. To address this, each video frame is split into smaller subframes before transmission, so that the loss of a single packet affects only a portion of the frame rather than the entire image. Considering

a video frame with a resolution of 1024x768 pixels. This frame is divided into 16 subframes by cutting it into a 4x4 grid—four slices along the x-axis and four along the y-axis. The images are stored in RGB format, meaning each frame consists of three color channels (red, green, and blue). After dividing the original frame, each subframe becomes a smaller image of 256x192 pixels, retaining the three-channel structure. This fragmentation ensures that even if a packet containing part of the subframe is lost, only a small section of the frame is impacted, thereby preserving the overall quality of the video during transmission. In this approach the video frames are fragmented into multiple subframes and forwarded through the network.

4.5.2 Capturing and processing

First, the raw images are captured from the camera fitted on a mobile node in the ad-hoc network. The raw images are stored in RGB format, that is for color images three channel matrices are used. Next, the matrix of the raw image data set is divided into 4x4 sub-matrices, resulting in the creation of 16 smaller frames. Each frame consists of a 256 X 192, 3-layer image matrix. These image matrices are cached in temporary memory for further processing before sending. While sending, in the beginning, all the sub-matrix corresponding to the first frame are sent by placing every sub-frame first in the cache and then forwarding it to the network. From the 2nd frame onward, every current sub-matrix value is compared with the previous one. If the changes in the pixel values go beyond the threshold limit, then only the frames are prepared for transmission. As the frame data is large, the application compresses each frame in JPEG format before forwarding the data to the transmission channel. Before sending each frame (which is eligible for transmission), a frame ID is attached with the frame's data for recognizing the frame at the destination.

At the sink node, after receiving the image frames in JPEG format, the image sub-frames are decoded into matrix format again to display the frames individually. If there are no new frames arriving at the sink node, the application continues showing the old frame. When new frames arrive, the application replaces the old frames with the new ones.

As shown in Figure 4.1, the camera produces video frame *frame1* at time t_0, *frame2* at time t_1 and so on. Every frame is divided into 16 parts, $subframe_1$ to $subframe_{16}$. At the time of sending, the application calculates the pixel difference between a subframe in the current frame and the corresponding subframe in the previous frame. For example, as shown in Figure 4.1, while sending the frame1, pixel difference between $subframe_1$ of frame1 and $subframe_1$ of frame0 is calculated. Similarly, $subframe_2$ of frame1 and $subframe_2$ of frame0 is calculated and so on. If the n-th subframe of $frame_{t-1}$ is already transmitted from the sender, and it is found that the pixel difference between the $subframe_n$ of $frame_t$ and the $subframe_n$ of $frame_{t-1}$ crosses the threshold limit, then only the $subframe_n$ of $frame_t$ is transmitted. If the pixel difference is lower than the threshold, then the particular subframe is not sent by the sender. A global variable is used for storing the threshold value. The threshold value can be changed by the user to optimize the performance of an application. Thus, large-sized video frames are divided into small frames and transmitted based on significant changes in pixel values to reduce the data transmission over unreliable mobile ad-hoc networks.

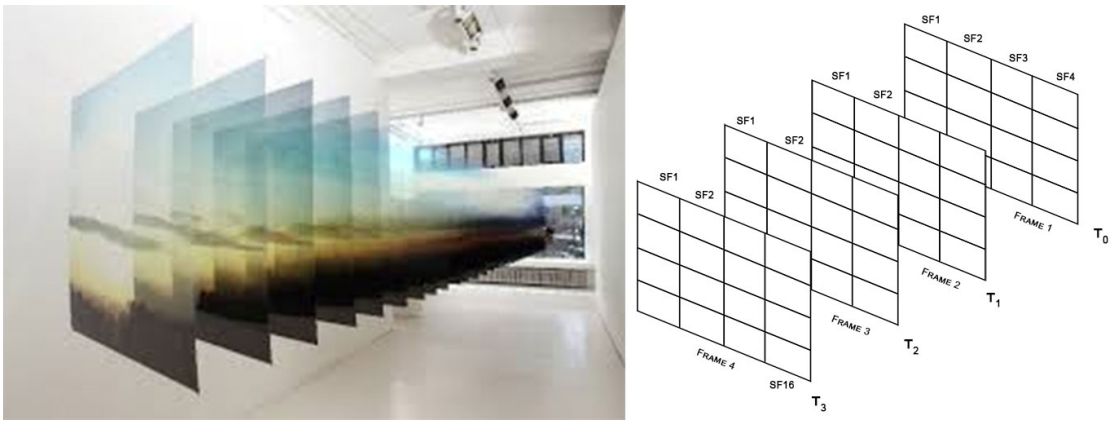

Figure 4.1: Cross layered Image frames

4.5.3 Releasing sub-frames to network channel

The application has three types of nodes for streaming the captured video. These nodes are

- **Source Node** This node captures the video frames with the help of the attached 'No Infrared filter' (NoIR) camera, processes the image frames, generates the sub-frames and sends the frame data to the 'send' queue. The 'Send' queue sends the data to the network interface module. The network interface module prepares the data packet by placing data as a payload.

- **Intermediate Nodes** There are multiple intermediate nodes which are responsible for forwarding the data from the source node to the sink node.

- **Sink node** Sink node receives the data packets and reconstructs the image subframes. The application at the sink node reconstructs the video frames from the incoming sub-frames.

To deliver the image data from source to destination, the application uses UDP server-client protocol.

On the client side of the application, which runs on the source node, the program continuously pushes data to the server using the server's IP address. There is a channel established using a defined port. The client code is actively sending video frames or image data packets to the server without waiting for an acknowledgment as it is running on UDP protocol The server side of the application runs on the destination node where the video is displayed. Figure 4.2 shows the application architecture. In this process, the Raspberry Pi's NoIR camera captures an image and sends the digitized image to the OpenCV library in CV_8UC3 format, which is a raw image format. Using OpenCV, the image is divided into 16 smaller subframes for processing. After the subframes are processed, a UDP client transmits the image data to BATMAN-adv, which then sends the data over a wireless network via the network card. The data is forwarded through multiple intermediate nodes using multi-hop communication, eventually reaching the base station or sink node. At the sink node, the process is reversed, i.e. the wireless network card receives the data, converts it into a digital format, and sends it to OpenCV to reconstruct the full video frames for display.

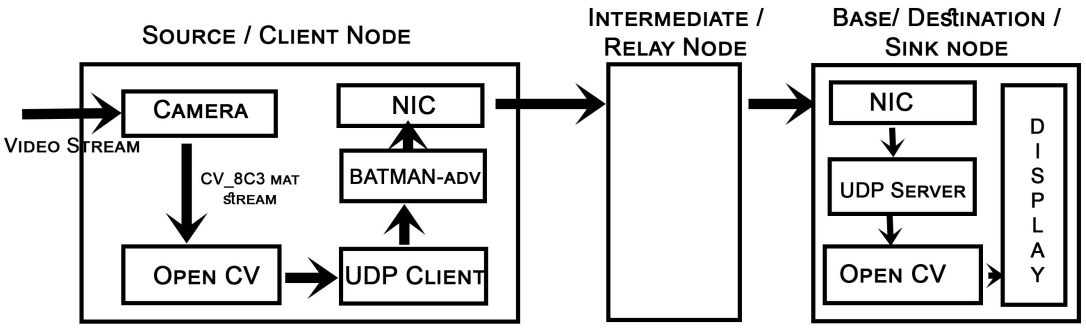

Figure 4.2: Overview of the application architecture

4.5.4 Algorithm for cross-layered video transmission

The methodology used in the research work is the testbed method. An algorithm for transmitting video over the MANET testbed is shown in Algorithm 1, which is developed on top of BATMAN-Adv. The functions shown in the algorithm are executed on the client node.

The algorithm is responsible for sending the captured video data to the server address port. The objective of the algorithm is to decrease the size of the video data which is required to be transmitted through the wireless channel. The functions of the algorithm are explained here using an example. Each frame of the captured video is stored in temporary storage after dividing into 16 sub-frames.

Initially when the application starts, i.e. at the instance of time t_0, the subframes of the first video frame vf_0, i.e. vf_{0_0} to $vf_{0_{15}}$ are stored and are also transmitted to the network channel. Some initial parameters are defined for this algotithm. The grid size had been initialized to 16, the threshold limit was initialized to 80% of pixel change between the previous and the current sub frame and the FPS for the NoIR camera was set to 30. At time t_1, when the next video frame vf_1 is captured, it is divided into 16 parts, svf_{1_0} to $svf_{1_{15}}$. The algorithm then computes the pixel differences Δ for each subframe δ_1 to δ_{15}.

δ_n = diff of svf_{n-1} and svf_n where 0 $\leq$ n $\leq$ 15

Algorithm 1 Algorithm for capture, process and transmit

Require: $videoCaptureCap(0)$
Ensure: $gridSize = 16$
  UDPSocket sock
  if *cap* is not open **then**
    Exit
  end if
  while *true* **do**
    if *cap* size =0 **then**
      continue
    end if
    $x \leftarrow \sqrt{gridSize}$
    $y \leftarrow \sqrt{gridSize}$
    $frameIdentity \leftarrow 1$
    $matrixArray[4][4] \leftarrow$ 16 zero matrix of size $(capRows/4)X(capCol/4)$
    while $x \geq 0$ **do**
      while $y \geq 0$ **do**
        $subCap \leftarrow$ submatrix of cap of size $(capRows/4)X(capCol/4)$
        $pixelDiff$
        if $compare(matrixArray[x][y], subCap) \geq thresholdLimit$ **then**
          $capEncode \leftarrow$ encode sub matrix to JPEG
          append $frameIdentity$ at the end of the $capEncode$
          $send$ capEncode to $servAddress, servPort$ via $sock$
        end if
        $y--$
        $frameIdentity++$
      end while
      $x--$
    end while
  end while

If the pixel difference of any subframe is greater than a threshold limit τ, i.e. $\delta_n \geq \tau$, then the sub-frame svf_n becomes ready for sending. Before sending, the raw subframe is compressed to JPEG format and the subframe ID i where i ≥ 0 and i ≤ 15 is appended to the compressed jpeg image data so that it can easily be recombined with the other subframes at the destination.

In Algorithm 1, there are two main operations which consume a significant amount of time. One is dividing the frames into sixteen parts and another is finding the pixel difference between the previous and current subframes. For the first operation, constant time is required. However, to check the differences, the

algorithm needs to iterate through the rows, and the columns for each row. The algorithm needs to calculate the pixel difference for all 16 subframes, where 16 is the value of the parameter 'gridsize' used in the algorithm. The value of the grid size may be changed depending on the video resolution and other parameters. Thus, the complexity is $O(n^2)$, where $n = \sqrt{gridSize}$ or $O(gridSize)$. In general, the algorithm needs to evaluate the pixel difference between all subframes for every frame.

Explanation of other data structure is given below:

- *videoCapture Cap(0)*: This is the variable where the raw video data is storing from camera feed. There is only one camera is installed in the SBC, so $Cap(0)$.
- *gridSize*: Total number of subframes
- *frameIdentity*: It store the count of the frames
- *matrixArray*: In this array, the pixel values of the captured single image frame is stored.
- *capRows*: Captures single image pixel value of X axis
- *capCol*: Captures single image pixel value of Y axis
- *capEncode*: Stores the JPEG format values of the subframe image
- *servAddress*: Stores the destination IP address
- *servPort*: Stores the port number of destination sever

Algorithm 2 runs on the server node and receives the frames, assembles and decodes them and displays them at the sink node. Algorithm 2 is responsible for assembling the subframes into the larger frame. A UDP socket buffer continuously receives data from the wireless network channel at the defined port address. The received subframe data $rsvf_n$ followed by the subframe ID n is checked by the application to find whether it is corrupt. If so, the application discards the subframe.

Algorithm 2 Algorithm for receiving, decode and display the video

```
UDPSocket sock
Char Array buffer                    ▷ *Larger than maximum UDP packet size
while true do
    buffer receive from servPort via sock
    extract the frameIdentity
    decodedMatrix ← decoded buffer
    Show the decodedMatrix by frameIdentity
end while
```

If the received subframe is valid, the main payload and the subframe ID are extracted from the received data.

At the receiver end, 16 blank containers are created with id 0 to 15. These containers c_0 to c_{15} are used to view the images inside it. The application replaces the old images with the new images depending on the subframe ID. If there are no new subframes transmitted for a particular container, the container retains the old subframe. Thus, at the receiver end, in the worst case scenario, the complexity will be O(gridSize)

i.e. if $(rsvf_n)$: $c_i \ll rsvf_n$ where $0 \leq$ i ≤ 15.

This means that only a small number of subframes are received in the image container. As a result, there is no noticeable movement in the video, and no significant pixel changes occur.

4.6 Storage and retrieval of cross-layered streaming video

The cross-layered approach is developed for high-definition video streaming using the low bandwidth wireless channel. Using this approach and with the help of BATMAN-Adv multi-hop mesh wireless network, it is now possible to gain visual access to remote areas. But at the same time, the streaming videos are needed to be stored for future inspection. In this section, an approach to the storage and retrieval of high-definition video is presented.

Algorithm 3 Algorithm for storing the streaming video

starts receiving
while *true* **do**
buffer receive from *servPort* via *sock*
Extract the *frameIdentity*
Extract the *subFrameIdentity*
Store the *subFrame*
Save the *subFrameIdentity* and *currentTimeStamp* to the key of *frameIdentity*
end while

4.6.1 Storage

Algorithm 3 is describes the approach to store the streaming data. The storage application continuously fetches the frame data from the temporary buffer. Every frame has the information of the frame ID and subframe ID associated with it. The application extracts the meta-information from the frames and stores the frames. The meta-information is now saved as a key-value pair, where the key is the frame identity or frame ID and the value is an array consisting of the subframe identity and the timestamp.

Suppose, at time t_t, two video subframes svf_x and svf_y $(0 \leq x, y \leq 15)$ of frame f_n are needed to be stored. As per the algorithm, subframe identity x and y and frame identity n are extracted and svf_x and svf_y are saved in non-volatile storage and the information is stored as key-value pair array n {frame:{x,y} timestamp: t_t }. This process continues till the video streaming is active. The proposed structure for storage is as follows:

```
"frameIdentity": {
  {
      "counter": "867",
      "subFrameIdentity": [
      "2","6", "7", "15"
      ],
      "currentTimeStamp": "Current Time Stamp"
  },
  {
      "counter": "868",
      "subFrameIdentity": [
      "7", "8", "15"
```

```
        ],
        "currentTimeStamp": "Current Time Stamp"
    }
}
```

The structures of the two frames (867, 868) are shown above. In frame #867, only #2, #6, #7, #15 subframes will be stored and for #868 frame, subframe nos #7, #8, #15 will be stored. While displaying the frame, these subframes will replace the corresponding previous subframes to make a complete frame. In the worst case, when there are changes in every subframe, all the subframes in every frame need to be stored and the space complexity increases to $O(mn)$, where m is the number of frames per second and n is the number of subframes in a frame. However, in practical situations, the space required for storing the frames is very low compared to other approaches. The algorithm creates a storage space of size n to store the input elements. The application stores the frame as it comes, so the space complexity is $O(x)$, where x is the incoming video subframes.

4.6.2 Retrieval

The stored video is needed to be retrieved while playing the video. Algorithm 4 is implemented to retrieve the video.

The retrieving process creates sixteen blank frame containers for displaying the images. The application starts reading data, $frameIdentity$ from buffer A memory and places all the subframes vf_{0_0} to $vf_{0_{15}}$ of the very first frame vf_0 in the blank containers c_0 to c_{15}. Now the application continues reading the next $frameIdentity$ value (say n) and replaces the previous subframe x with the new subframe vf_{n_x} in the containers c_x. This process continues for all subframes stored with the $frameIdentity$ n. While expressing how the runtime of the retrieval algorithm grows as the input size increases, it has a limit of max subframes to retrieve. In each second, the maximum number of subframes should be 16 and and the minimum number of subframes can be 0 (when there is no significant difference in pixel values and no frame was transferred).

Algorithm 4 Algorithm for retrieving the stored video

```
Start retrieving
Copy the subFrame into bufferA
Copy the key of frameIdentity and values into bufferB
Generate display container C0 to C15
Set counter to 0;
while true do
    Read the value of frameIdentity [ counter ]
    Fetch identities of the subFrame from frameIdentity [ counter ][ subFrameIdentity ]
    Fetch subFrame from buffera
    Set container to 0
    while container <15 do
        if Is exists frameIdentity [ counter ][ subFrameIdentity ][ container ] then
            Display frameIdentity [ counter ][ subFrameIdentity ][ container ]
        end if
        Increment container by 1
    end while
end while
```

4.7 Implementation of test-bed using Raspberry Pi

For implementation of the above algorithms, a tesbed was created with 16 Raspberry pi 3B+ nodes. A working Raspberry Pi 3B+ is displayed in Figure 4.3, each with 1GB LPDDR2 RAM with a 1.4GHz 64-bit quad-core processor. The operating system used in this implementation is Raspberian OS, released on May 7th 2021, kernel version 5.10. The destination(sink) node is a Rasbberry Pi 4 node consisting of Quad core Cortex-A72 (ARM v8) 64-bit SoC 1.8GHz Processor with 8GB LPDDR4-3200 SDRAM. The destination(sink) is a fixed node. The Figure 4.4 shows the image of Raspberry Pi 4. A monitor is attached with the sink node for viewing the streaming video. Figure 4.5 is showing the image of Raspberry Pi monitor. The other nodes, including the source node, are mobile nodes. 4WD robot cars are used for the intermediate nodes. Figure 6.7 shows structure of the mobile nodes. One NoIR(No Infrared) camera is attached to the source node for capturing the images and/or videos. Figure 4.7 shows the structure of a NoIR Raspberry Pi camera.

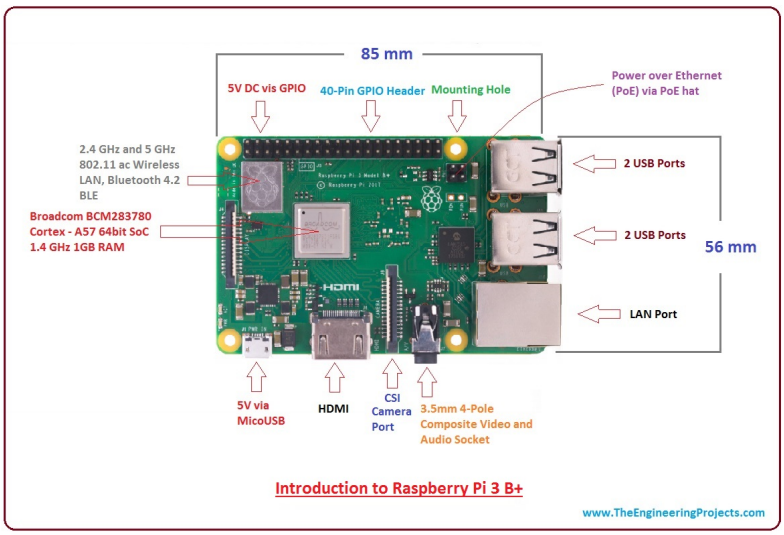

Figure 4.3: Raspberry Pi 3B+

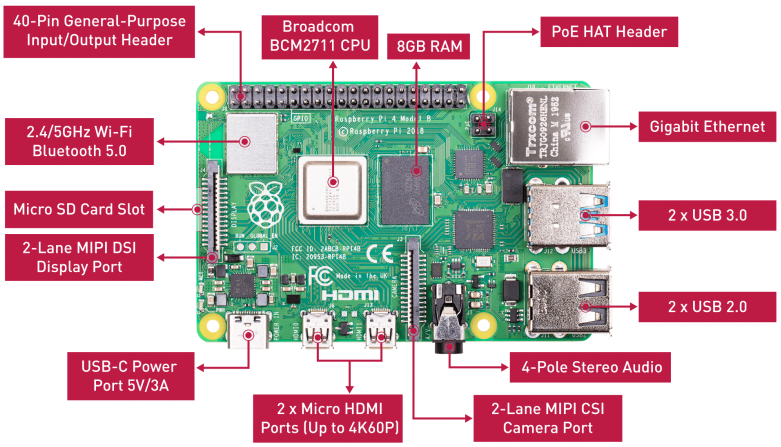

Figure 4.4: Raspberry Pi 4

Each node in the MANET is attached to a wireless transceiver. All nodes are connected in a mobile ad-hoc network configured with BATMAN-adv and forming a mesh network, in which the best routing path is calculated by the routing algorithm used by BATMAN-adv.

4.7.1 Overview of BATMAN-Adv

The BATMAN (Better Approach To Mobile Ad-hoc Networking) protocol was designed to improve on OLSR (Optimized Link State Routing) protocol by handling unreliable links in ad-hoc networks. Each node maintains information about the best next hop toward other nodes, eliminating the need for global knowledge of the

Figure 4.5: Raspberry Pi display monitor

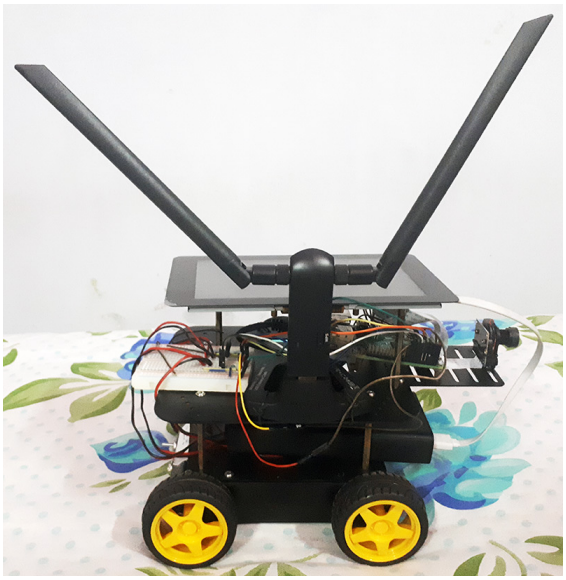

Figure 4.6: Replica of intermediate mobile node

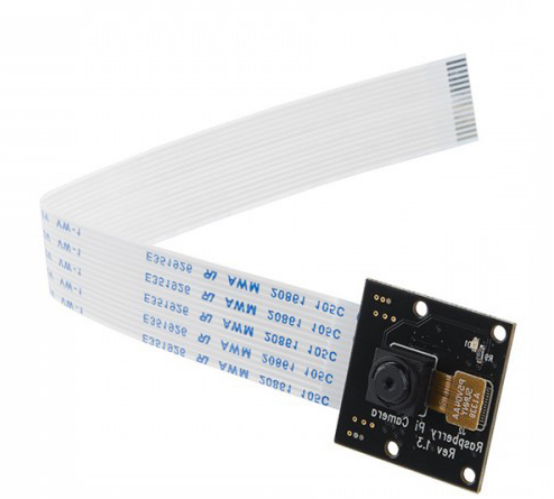

Figure 4.7: Raspberry Pi NoIR camera

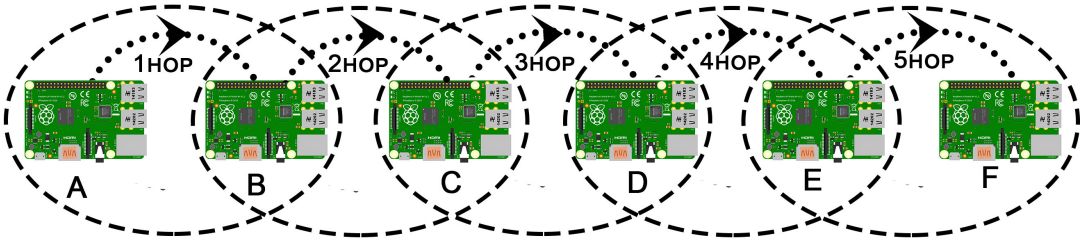

Figure 4.8: Multi-hop network topology

network topology. Initially implemented as a layer 3 routing protocol using UDP, BATMAN evolved into BATMAN-adv, a layer 2 protocol in the Linux kernel that uses MAC addresses for neighbor communication and emulates a virtual network switch. BATMAN-adv v4 introduced a Transmit Quality (TQ) mechanism to handle asymmetric links, but it struggled with growing network sizes, leading to v5, which uses a throughput-based metric. BATMAN operates as a proactive distance-vector routing protocol, periodically sending Originator Messages (OGMs) to check link existence rather than quality. The nodes of the BATMAN, only track the best next hop, and the TQ metric helps optimize routing by counting received OGMs within a sliding window [76].

The topology of the multi-hop ad-hoc network used for this work is shown in Figure 4.8. Here, node A is set as the sender node for the video streaming application

and the far-right node is set for receiving the streaming video. The nodes can move randomly at the rate of 7km/hr and nodes are placed in such a way or at such a distance that no direct line-of-sight communication path exists from source to destination. The data should travel from source to destination by hopping through multiple relay nodes or intermediate nodes. By changing the number of the relay nodes, the video streaming algorithm is implemented with different number of hops.

4.7.2 Establishing communication

All nodes(Raspberry Pi) are equipped with the built-in WIFI dual-band 802.11 ac wireless LAN (2.4 GHz and 5GHz). BATMAN-adv is installed and configured in every node for making a mobile ad-hoc (MANET) testbed. The protocol uses 5.2 GHz (channel 40) for ad-hoc communication. The BATMAN-adv is responsible for establishing the communication channel between the nodes, calculating the best path from the source to the sink, and finally, transmitting the data to the sink node. Figure 4.9 shows the mobile ad-hoc mesh network built by BATMAN-adv.

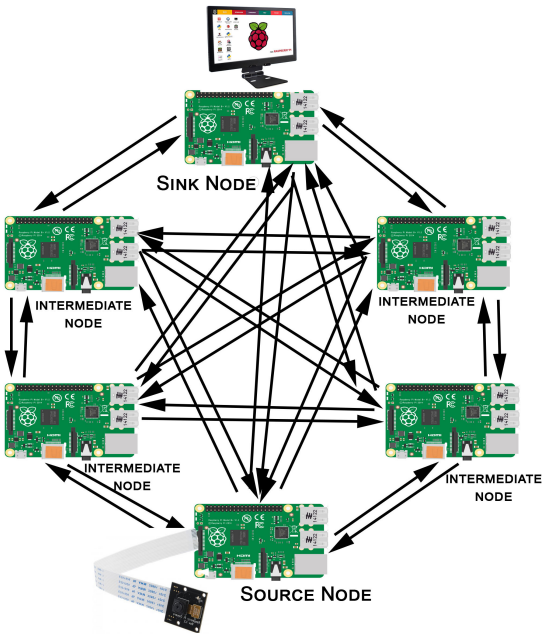

Figure 4.9: BATMAN Mesh network

For experiment purposes, the mobile nodes are placed with such an orientation that some parallel paths exist in the network.

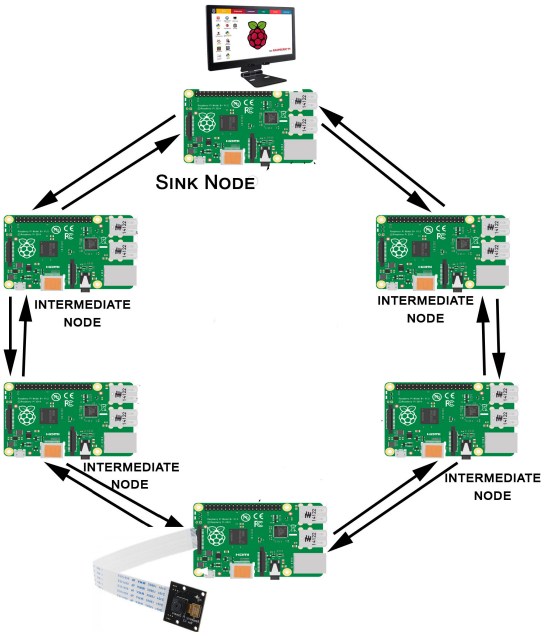

Figure 4.10: Replica of intermediate mobile node

The Raspberry Pi NoIR captures 30 image frames per second with the resolution of 1024 X 768 pixel, a pixel representing RGB colour pixel value stored as 24 bit as follows.

$R = 0 - 255 \implies 8$ bit

$G = 0 - 255 \implies 8$ bit

$B = 0 - 255 \implies 8$ bit

Raw images from the Raspberry Pi NoIR camera are obtained by accessing the 'video0' file using OpenCV open-source library using C++ code. As the operating system is ubuntu, the input and output ports are treated as files. This concept is part of the Unix philosophy that "everything is a file. Now, OpenCV composes colours using the BGR colour space (red and blue channels are swapped). For color images, three-channel matrices are used. defined as CV_8UC3. where

CV_ <bit_depth>{U|S|F} C<*number_of_channels*>

CV: OpenCV library

U: unsigned integer

S: signed integer

F: float type

C: channel

This way CV_8UC3 matrix is formed.

The smallest data type possible is char, which means one byte. This may be unsigned (so can store values from 0 to 255) or signed (values from -127 to +127). When three components are used (as in the case of RGB), it gives 16 million possible colours.

The size of a raw image frame of size 1024 X 768 pixels is nearly about 2.4MB (1024 X 768 X 24). As the camera captures 30 images in one second, 72 MB of data is generated in one second for streaming. This means nearly 75 MBPS bandwidth is required to transmit the data through the unreliable ad-hoc channel. Because of the unreliability and unpredictability of the channel, packet loss can occur frequently and the packet losses can destroy the entire frame of the video. While conducting the experiment using proposed algorithms it has been observed that in a multi-hop environment, after 3 to 4 hops, the FPS (Frames per second) drastically reduces to 1 to 2 frames.

4.7.3 Dataset details

For the experiment of video streaming over mobile ad-hoc Networks, the dataset was generated using the experimental set up discussed above. Real time videos were captured and transmitted over MANET. Some of the key aspects regarding the dataset are discussed below.

- **Source of the dataset:** The application uses NoIR cameras integrated with Raspberry Pi. This camera has no Infrared filter. This means that pictures the camera takes in daylight will look bright, but it gives the ability to see in the dark with infrared lighting. This camera will capture 20 to 30 image frames per second. These images are stored in a RAW and uncompress format in three-channel matrices. This corresponds to the RGB color space, where each pixel contains three values: one each for red, green, and blue. Each pixel is represented by three 8-bit values, one for each channel, 3 bytes in memory. The image data is stored in a contiguous block of memory. The rows are

stored one after the other, and each pixel in a row consists of three consecutive bytes for the RGB channels. If the image has a width of W pixels, a height of H pixels, and is stored in the CV_8UC3 format, the total memory required will be W * H * 3 bytes. Thus, this is loss less and take lot of memory space to store an image data.

- **Data collection process:** The raw images are stored in a buffer located in the local storage after capturing the images through the camera.
- **Characteristics of the dataset:** The image data are stored in the local storage. Every second the camera captured 72 MB of data. The detailed description is already discussed in the Section 4.5.2.

While collecting the dataset, the unique characteristics of MANETs, such as the dynamic nature of the network, limited bandwidth, and available resources, have always been kept in mind. This can affect both the collection and analysis of the data and should be taken into consideration while interpreting the results.

4.8 Implementation of the algorithm

The algorithms are implemented using C and C++ languages. For capturing and dividing the image frames, the OpenCV C++ library is used. OpenCV is an open-source C++ library for image processing and computer vision, developed by Intel, and now maintained by Itseez [45]. After dividing the images, the application calculates the difference between two-pixel matrices (previous and current subframes) made by the OpenCV library itself. The threshold value is set to 80%, i.e. if the pixel difference between the previous and current subframe is more than or equal to 80%, then only that subframe will be transmitted. For transmission, the UDP protocol is used. The UDP socket is designed and written using C++. The application sends the image frames from the client to the server. The client is the source and the server is the destination. The application needs a server / destination IP and port for sending. The intermediate path is managed by the BATMAN-adv protocol.

4.8.1 Experimental results for streaming data

With the setup described in the above subsections, the proposed Algorithm 1 is implemented on the client node for transmitting video over the MANET testbed developed with BATMAN-Adv. When the application streams the image sub-frames using the proposed algorithm, only one sub-frame is transmitted, when the contents of the other sub-frames do not differ from the previous frame contents. In this scenario, only 0.14MB of data is required to be sent. Moreover, the application encodes the subframes to reduce their size. In the other cases as well, only a little amount of data is required to be transmitted, and therefore the packet loss is also decreased.

The experiments are carried out to demonstrate the efficiency of the proposed algorithms. In this work the experiments have been carried out in an indoor environment. Four nodes have been placed as shown in Figure 4.13. The other nodes have been placed in an indoor environment such a way that some parallel paths exist between the source and the sink. Here, Node 1 is the source node and Node 4 is the sink node.

```
pi@raspberrypi-102:~ $ sudo batctl p -R  dc:a6:32:0e:e1:db
PING dc:a6:32:0e:e1:db (dc:a6:32:0e:e1:db) 116(144) bytes of data
116 bytes from dc:a6:32:0e:e1:db icmp_seq=1 ttl=48 time=89.08 ms
RR:     b8:27:eb:ad:aa:ce
        dc:a6:32:47:29:2c
        b8:27:eb:31:85:90
        dc:a6:32:0e:e1:db
        dc:a6:32:0e:e1:db
        b8:27:eb:31:85:90
        b8:27:eb:ad:aa:ce
```

Figure 4.11: Demonstration of parallel path by BATMAN command

The BATMAN ping command was used to verify the available path between the source and sink nodes, as shown in Figure 4.11. The MAC address of the source node (node-1) is ‘b8:27:eb:ad:aa’, and the MAC address of the sink node (node-4) is ‘dc:a6:32:0e:e1’. When the ping request was sent, the packet followed the route: from ‘b8:27:eb:ad:aa’ (node-1) to ‘dc:a6:32:47:29:2c’ (node-2), then to ‘b8:27:eb:31:85:90’ (node-3), and finally to ‘dc:a6:32:0e:e1’ (node-4). However, during the response, the packet took a different parallel route to return from the sink to the

source. The response path was: from 'dc:a6:32:0e:e1' (node-4) to 'b8:27:eb:31:85:90' (node-3) and then directly to 'b8:27:eb:ad:aa' (node-1). Figure 4.12 shows the pictorial representation of parallel path.

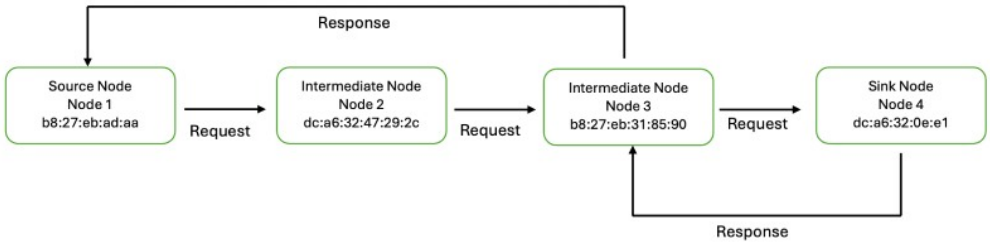

Figure 4.12: Demonstration of parallel path

For the result, the FPS (Frames per second), RTT (Round trip time), packet loss and bitrate are observed. Each time the experiment has been carried out for 20 to 30 minutes and the resulting data set are generated by measuring FPS and RTT. The average of FPS, as well as RTT have been computed for a particular experimental setup. The results obtained from the proposed algorithm are compared with the result obtained from the implementation of a single-layered algorithm.

Figure 4.13: Indoor orientation

The result set is given in the Table 4.1 and 4.2.

For better understanding, the experimental data of single-layered and cross-layered algorithms are plotted graphically in the same 2D plane as shown in

Table 4.1: Experiment result-set for cross-layered video stream

Distance (ft)	Hop count	FPS	RTT (ms)	Packet Loss (%)
0	1	30	1.02	0
15	1	25	2.75	0
34	1	23	38.75	0
42	1	21	119.79	40
54	1	20	25.69	10
60	2	23	38.27	0
65	2	14	146.28	0
68	2	10	187.91	20
70	3	16	40.92	10
74	3	9	108.31	35
80	3	5	191.32	46
95	4	7	68.32	14
100	4	7	112.45	30

Table 4.2: Experiment result-set for Single layered video stream

Distance (ft)	Hop count	FPS	RTT (ms)	Packet Loss (%)
0	1	30	1.13	0
16	1	24	2.36	0
40	1	21	10.87	0
45	1	19	45.71	10
59	1	19	48.96	0
65	2	17	24.65	5
69	2	14	60.38	20
70	2	10	63.54	30
75	3	5	54.19	34
78	3	5	73.74	35
85	3	3	98.30	46
98	4	2	104.73	20
110	4	1	129.21	87

Figure 4.14, Figure 4.15 and Figure 4.16 . It is observed that in both cases, the FPS value falls when the hop count (Figure 4.14) and the distance between the sink and the source node (Figure 4.15) increase. The FPS reduces because of the network packet loss. When distance increases, one needs to put one or multiple nodes in between the source and sink nodes so that the communication channel is not broken. If the intermediate node count (hop count) increases, the resulting FPS value decreases because of the network packet loss. Keeping this issue in mind, when cross-layered video transmission is used, a higher FPS value is achieved

compared to the single-layered video transmission algorithm. In the case of single-layered transmission, the entire image frame will be destroyed if the lost packet contains any frame information. The application will not be able to construct the image frame as the application is not getting all the information required for construction of the frame. So if the packet loss increases, the FPS also decreases. In case of cross-layered video transmission, the frames are divided into smaller parts. Therefore, the lost information can hamper only the smaller parts of the large image frames. The other smaller parts can easily be used to construct the large image frame with the previously received frame of the lost part.

It can be observed from Figure 4.14 that at 4 hops, single-layered approach receives 1 FPS, whereas, cross-layered approach receives 7 FPS. This type of gain in FPS can also be observed when distance increases (Figure 4.15). When the experiment was performed with a distance of 100-110 ft, 3 intermediate nodes were required and hop count became 4. In this situation, the FPS is 1 and 7 for the single -layered and cross-layered cases respectively. The distance covered in the experiment shown in Figure 4.15 is 110 feet when the hop count is 4, and in the case of Figure 4.15, 3 intermediate nodes have been added to cover more than 100 feet.

The above experiments have been carried out using Raspberry Pi 3b+ and 4. The internal built-in antenna has been used for this experiment. If external long range antenna could be attached, the range could be much larger compared to the range obtained here.

4.8.2 Experimental results for storing data

Storing of the streaming data is also necessary for historical data validation. For this experiment, the video is captured and stored in a single-hop network. For the experiment purpose, the captured video quality is kept very high (Full HD video 1920 X 1080) and before sending the video frames, quality has been reduced to 25%. As the experiment is done in one hop, it was possible to keep the constant FPS (30) at the capturing end and the receiving end. The experiment is done with single-layered video transmission and cross-layered video transmission. In this

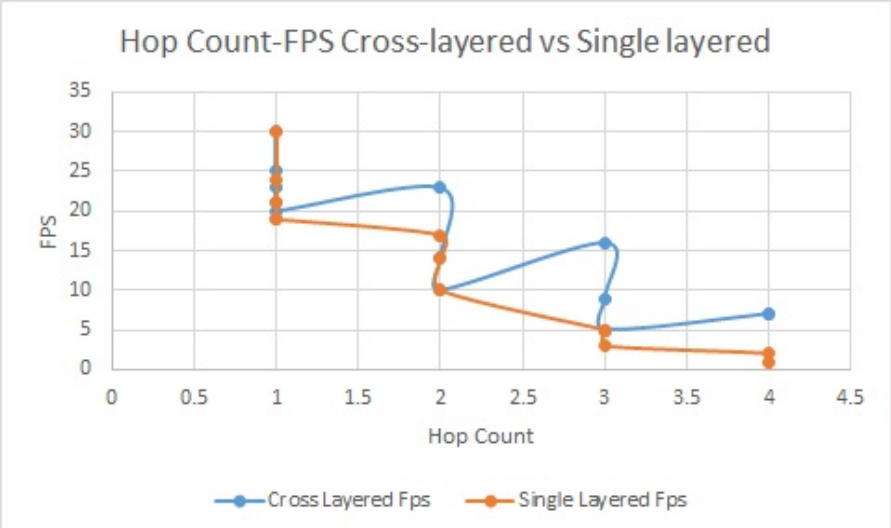

Figure 4.14: Hop count vs FPS in both cases

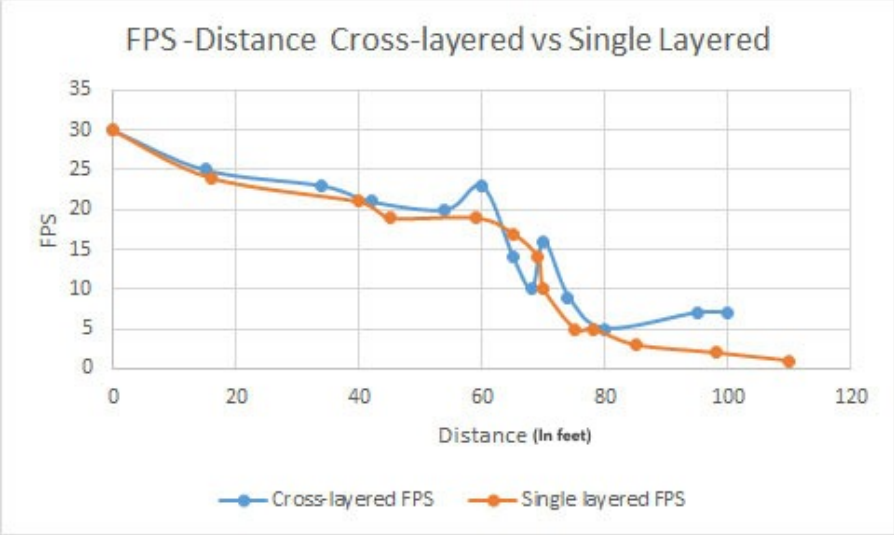

Figure 4.15: Distance vs FPS in both cases

experiment 16 minutes of video has been captured for both cases. The experimental results are shown in Table 4.3

After plotting the data (Figure 4.16), it is observed that the cross-layered video takes much less storage space compared to the single-layered. For example, for a 16 minutes video, where the cross-layered approach needed 831.52 MB of storage, the single-layered approach took 4.93 GB of storage space. From the experimental results, it is also noticed that in the case of the 16-minute video, at least 14% of storage reduction is achieved and it is also demonstrated that maximum 23%

Table 4.3: Experiment Result-set for Single Layered and Cross Layered Video Storage

Time	Single Layered	Cross Layered	Reduction (%)
1 Sec	5.23 MB	1.175 MB	22 %
10 Sec	51.219 MB	7.121 MB	14 %
30 Sec	152.129 MB	31.3 MB	20 %
1 Min	293.5 MB	52.124 MB	17 %
2 Min	576.2 MB	133.32 MB	23 %
4 Min	1.33 GB	215.2 MB	16 %
8 Min	2.37 GB	499.5 MB	21 %
12 Min	3.89 GB	764.13 MB	19 %
16 Min	4.93 GB	831.52 MB	17 %

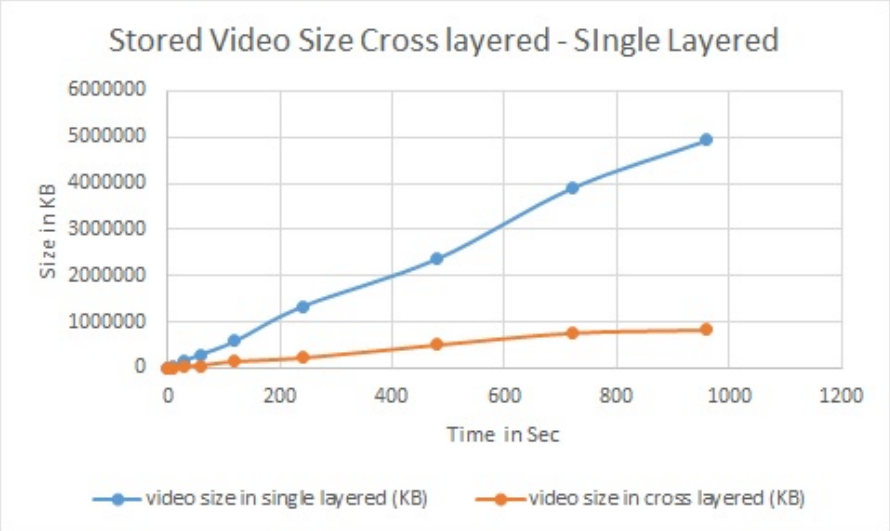

Figure 4.16: Time vs Size in both cases

of storage reduction is achieved. More reduction can be achieved if there is no movement in the captured surrounding environment.

4.9 SMS-based health-care delivery

Taking Health-care services to the door step of the rural people requires special attention. The communication infrastructure in these locations is usually poor compared to urban areas. Access is the major issue in rural health around the world, even though the majority of the population lives in rural areas. The resources are concentrated in the cities. The people of the rural and remote areas have difficulties with electricity, transport, communication and for that reason, they

all face the challenge of shortages of doctors, medicines and other resources. To overcome the primary health-care obstacle in the rural and remote area, kiosk based health centers **KiORH** (**K**iosk **O**perated **R**ural **H**ealthcare) have been setup in the rural areas with the help of medical professionals using Android mobile phones, tablets, medical sensor devices. KiORH runs on Android based mobile gadgets and connects to cloud server, thereby creating an integrated environment for the patients and remotely located medical professionals to interact in comprehensible way. Treatment episodes are created to deliver basic health-care services to the rural patients. However, the key challenge for provisioning such remote health-care system is interruption or intermittent disconnection in telecommunication or broadband services in rural areas. Therefore, a unique feature is added to KiORH by which the application at a rural kiosk can efficiently communicate with the application running at the computing device owned by the urban doctors via Short Message Service (SMS), particularly at the time of unavailability of Internet connectivity. The rural health workers check up the visiting patients and collect the patients' symptoms and complaint-related data and transmit to the doctor. The doctor, on the other side of the system (probably in urban area), analyses those data on real-time basis and generates prescription or puts forward advice related to the treatment of the patient at the urban kiosk.

4.9.1 Application work flow

The KiORH application workflow was discussed in Section 3.3.1. After collecting all details, the application sends the data to the cloud through the Internet. But, if there is irregular connectivity, the application sends the patients' vitals to the cloud using SMS instead of using the Internet. The application takes the important patients' data (particularly symptoms and vitals) in a JSON string, compresses the string and sends via SMS to a SMS server. SMS server sends the compressed data to the cloud server. Another cloud application decompresses the data and changes it to the original JSON format. The SMS sending screen is shown in Figure 4.17 and the flow is shows in Figure 4.18.

Figure 4.17: SMS Sending Screen

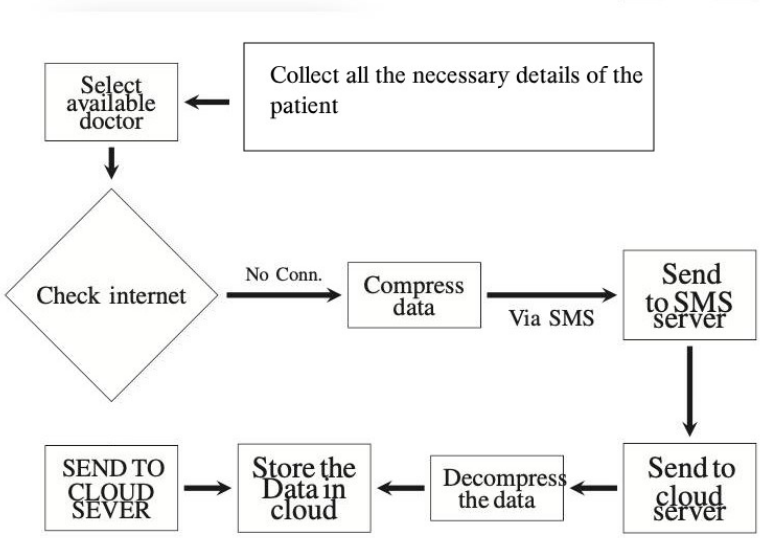

Figure 4.18: KiORH SMS sending work flow

A third party SMS service Twilio is used as a middle-ware to establish connection between Kiosk and Cloud server through SMS.

4.9.2 Compression method

Uncompressed data can take up a lot of space and character. In case of KiORH, the patients' vitals and other details require large number of bits and sending such data through SMS can become practically impossible due to its limited character specification. Hence, the health data needs to be compressed in order to send most of the information to the Doctor.

There are two categories of compression techniques - one is lossy and the other is lossless compression. Both techniques look for duplicate data, eliminates them

and makes a compact representation. As the name suggests, no information will be lost in case of lossless compression. It reduces its bits by identifying and eliminating statistical redundancy. On the other hand, lossy compression reduces bits by removing unnecessary or less important information. In case of lossless compression, after decompression, the entire information can be retrieved though that is not the case for lossy.

Lossy compression methods include DCT (Discreet Cosine Transform), Vector Quantisation and Huffman coding while lossless compression methods include RLE (Run Length Encoding), string-table compression, LZ77, LZ78 (Lempel- Ziv), LZW (Lempel Ziff Welch) and zlib. There exist several compression algorithms as mentioned above. However, in the current implementation, LZ77 and LZ78 have been used. The algorithms are described below.

- **Lempel-Ziv Algorithms LZ77:** LZ77 (LZ1) and LZ78 (LZ2) are the two lossless data compression algorithms published in papers by Abraham Lempel and Jacob Ziv in 1977 [126] and 1978 [127]. LZ77 and LZ78 are both theoretically dictionary coders. LZ77 maintains a sliding window over previously seen characters during compression, the decompression must always start at the beginning of the input. This was later shown to be equivalent to the explicit dictionary constructed by LZ78. Conceptually, LZ78 decompression could allow random access to the input if the entire dictionary were known in advance.

  The LZ77 algorithm achieves compression by replacing repeated occurrences of data with references to a single copy of previously encountered data. These repetitions are encoded as pairs of numbers, known as length-distance pairs. Each pair indicates that the next 'length' characters match the characters located 'distance' positions earlier in the uncompressed stream. The data is stored in a structure called a sliding window, which is why LZ77 is sometimes referred to as sliding-window compression. Figure 4.19 illustrates the LZ77 structure.

A phrase T_j starting at a position i is encoded as a triple of the form (distance, length, symbol). A triple (d, l, s) means that: $T_j = T[i...i+l] = T[i-d...i-d+l]s$ The string $T[i...i+l]$ of length l has another occurrence d positions earlier in the text. A decoder can simply copy the string from the already decoded part of the text. The values d and l should satisfy $d \in [1...dmax]$ and $l \in [0...lmax]$. That means the earlier occurrence should not be longer than $lmax$ and should start within a window $T[i-dmax...i-1]$. The algorithm searches the window for the longest possible match under the above constraints, i.e., it tries to maximize l.

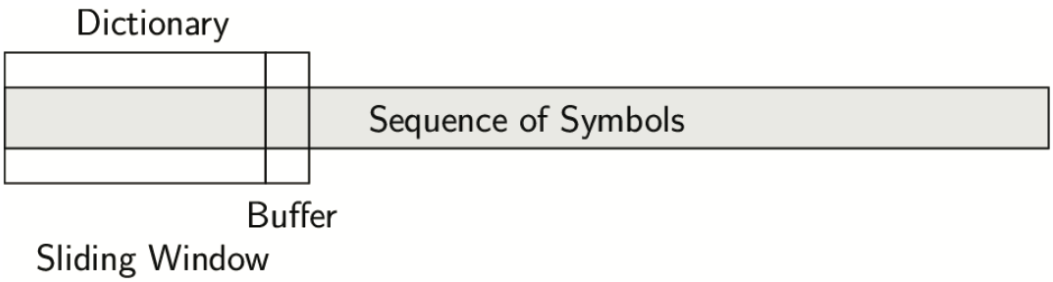

Figure 4.19: LZ77 Structure

- **Lempel-Ziv Algorithms LZ78:** Lampel-Ziv 78 or LZ78 lossless data compression algorithm has been published in 1978 by Abraham Lampel and Jacov Ziv. The algorithm, LZ78, is based on the same general idea, but the implementation is quite different from the earlier LZ77. LZ78 works by constructing a dictionary on the fly and the whole of it does not need not to received first before going to decompress it. For better understanding, an example is shown below. The string to be compressed is:

$$AABABBBABAABABBBABBABB$$

As per the algorithm, the shortest phrase at the beginning is considered, that will always be a single letter. In this case the letter is 'A'.

$$A|ABABBBABAABABBBABBABB$$

Now the next smallest phrase, which has not occurred earlier, is considered. In this case, it is 'AB'.

$$A|AB|ABBBABAABABBBABBABB$$

The next phrase 'ABB' has not occurred earlier, so 'ABB' will be the next phrase.

$$A|AB|ABB|BABAABABBBABBABB$$

In this way, after parsing the string, it can be as follows:

$$A|AB|ABB|B|ABA|ABAB|BB|ABBA|BB$$

Now, at the time of sending, instead of sending the entire string, the dictionary will be added with the phrase and sent as in the following structure.

1	2	3	4	5	6	7	8	9
A	AB	ABB	B	ABA	ABAB	BB	ABBA	BB
ϕA	1B	2B	ϕB	2A	5B	4B	3A	7

The first row contains the sequence numbers of the phrases, the second row is for the separated phrases and the third row contains their encoding. That is, when ABA is encoded from the fifth phrase, it is encoded as 2A. This maps to ABA since the second phrase was AB, and A was to it. Here, the empty set ϕ is considered to be the 0'th phrase and encoded by 0. Now, it is converted to binary form, mapping A to 0 and B to 1, the structure will be like:

$$,0||1,1||10,1||00,1||010,0||101,1||100,1||011,0||0111$$

Here 2 is mapped to 10 and 010, as the reference of the phrase invokes k bits (i.e. starting with $2^{k-1}+1$) dictionary elements, hence k bits is used to encode the phrase. The finally compressed binary string without separator will be

$$01110100101001011100101100111$$

To decode / decompress, the decoder needs to construct the same dictionary. To construct the dictionary, the first divider comes after 1 bit, the next will

come after 2 bits. For next two phrases, each separator will come after 3 bits. In general the separation will occur for 2^k of length k+2 bits. For example, the dictionary for above string is shown in Table 4.4

Table 4.4: LZ78 Decode Dictionary

ϕ	A	AB	ABB	B	ABA	ABAB	BB	ABBA
0	1	2	3	3	5	6	7	8

4.9.3 Data exchange

KiORH is generally implemented in the rural health kiosk where Internet bandwidth is very low compared to the requirements of the application for sending the patients' vitals to the cloud. So when the application finds that the bandwidth is lower than the threshold limit, it tries to send the health data through SMS. The currently acceptable SMS size allows not more than 160 characters. So the application extracts the clinically important information from the gathered data and compresses it using LZ78 algorithm. After successful gathering of all data from a patient, the application generates a JSON array which contains all details including patient's demographic info, current height, weight, pulse, body temperature, blood pressure, symptoms, habits, family history, current image, previous clinical documents (such as prescription and history followed by clinical examination records). The JSON is used because it is easy to convert to Javascript objects and send to server and vice-verse. Here is an example of the complete JSON string of a patient who complained about headache.

```
{
    "patientInfo"    : {
        "type"   : "existing",
        "reg_no"     : "000564",
        "name"   : "M␣Alok",
        "gender"     : "Male",
        "dob"    : "23-12-1980",
        "ph"     : "99993393",
        "occu"   : "service",
        "address"    : "Barrha,␣W.B."
        },
        "sensorData"     :{
```

```
        "height"     : "167 cm",
        "weight"     : "60 kg",
        "bmi"    : "21.51",
        "spo2"   : "94",
        "temp"   : "98.9"
        },
    "symptoms":[{
            "q" : "Duration",
            "a" : "2 days"
        },
        {
            "q" : "Where did it Start?",
            "a" : "Front right",
        },
        {
            "q" : "Where is it Now?",
            "a" : "Front right",
        },
        :   :   :   :   :   :   :
        {
            "q" : "Relieved By",
            "a" : "After taking medicine"
        },
        {   "q" : "Any associated symptoms", "a" : "None"},
        {   "q" : "Any Other comments", "a" : "None"}
    ],
    "previousDecoments":{ "previousXray" : "xray.jpg", "previousUsg"
:"usg_1.jpg",
    "previousLabReport":["bloodReport.jpg", ...]
    }
    "medicalHistory":[ "HighBloodPressure", "Stroke",
    "Asthama",
        :   :   :   :   :
    ],
    "habit":["Smoking","Tobaccochewing", ...], "familyHistory":{
    "father":["High Blood Pressure","Asthma"], "mother":["Diabetic","S
        :   :   :   :   :   :
    },
    "generalObservation":["In Pain","Looks unwell"], "clinicalExam":{
    "generalExam":[{
    "q" : "Check Eyes for Jaundice", "a" : "Normal" },
    {   "q": "Is there Eyes pallor?" "a" : "No"},
    :   :   :   :
    ],
    "faceExam":[{
```

```
    "q" : "Face: Swollen Face?",
    "a" : "Normal"}],
    :   :   :   :
    }
}
```

This whole string size is more than 3000 characters and it is impossible to send this information through SMS. Thus, it is required to extract the minimum required information from the above text using which the doctor from the other side can diagnose the cause of the patient's complaints. The application then extracts patient's name/ id (if the patient is a recurrent patient (RP)), current symptoms, current sensor data, recent report in textual format and current clinical examination parameters. The extracted JSON text will be as follows:

```
    {
    "patientInfo"   : { "reg_no"    : "000564"}, "sensorData"
:{
    "height"    : "167 cm",
    "weight"    : "60 kg",
    "bmi"   : "21.51",
    "spo2"  : "94",
    "temp"  : "98.9"
    },
    "symptoms":[
        {
            "q" : "Duration",
            "a" : "2 days"
        },
        {
            "q" : "Where did it Start?",
            "a" : "Front right",
        },
        {
            "q" : "Where is it Now?",
            "a" : "Front right",
        },
        :   :   :   :   :   :   :
        {
            "q" : "Relieved By",
            "a" : "After taking medicine"
        },
        {
            "q" : "Any associated symptoms",
```

```
                "a" : "None"
            },
            {
                "q" : "Any␣Other␣comments",
                "a" : "None"
            }
        ],
        "generalObservation":["In␣Pain", "Looks␣unwell" ],
        "clinicalExam":{ "generalExam":[
            {
                "q" : "Check␣Eyes␣for␣Jaundice",
                "a" : "Normal"
            },
            {
                "q": "Is␣there␣Eyes␣pallor?"
                "a" : "No"
            },
        :    :    :    :
        ],
        "faceExam":[{
            "q" : "Face:␣Swollen␣Face?",
            "a" : "Normal" }],
            :    :    :    :
        }
}
```

There is a small JavaScript program module which checks the number of the extracted characters and sends decreased percentage as a return value. The module runs in the application every time just after extraction. With the help of the return value, the application tries to further decrease the number of characters at least by 60%. Now, the questions and answers are fixed for that particular patient for that visit. Therefore, the question part is omitted and answer are used with the patient's reference ID. Thus, the length is decreased by 80% from the previous length, as examined by the previously mentioned program module. The JSON string now looks like:

{“patientReg” : “000564”,“sensorData” :[“167”, “60”, “21.51”, “94”, “98.9”], “symptoms”:[“1.1.1” : “2 days”, “1.1.2” : “1.1.3”, “1.1.3” : “1.1.3”, ... “1.1.7” : “After taking medicine”, “1.1.9” : “1.1.4”, “6:1:1” : “1 hour”, “6:1:2” : “6.1.3”, ...], “generalObservation”:[

"1", "3"], "clinicalExam":["1.1" : "1.1.2", "1.2" : "1.2.1", ... "7.1" : "7.1.1"] }

Next, the application applies string compression algorithm LZ78. The algorithm compresses the string nearly about 70% as the above said program module calculates and return the percentage value. The collection, extraction and compression methods are executed at the client side only, i.e. inside the Android tablet or mobile phone. This way the character length of the health data of a particular patient is decreased from 3000 to 100. This compressed string is then sent to a third party SMS server [17]. The server provides a number, an API key and an API URL. After receiving the SMS which is to be sent to the provided number, the SMS server sends the SMS data to KiORH server by making a secure connection by an API key through an API URL. As this process executes in real time, the KiORH server generally receives the SMS data as soon as the application sends it. The back-end application immediately decompresses it and saves onto the back-end database. Whenever new data is saved in the database, the Doctor side of the KiORH application immediately detects it and sends a notification to the Doctor application screen. Doctor, from the urban location checks it, prescribes medication, advice and medical tests for that patient. In the same way, the application sends the advice, prescription to the application running on the android device at the rural kiosk from where it came. Figure 4.20 shows the flow diagram.

In order to understand the suitability of the proposed scheme as above for sending the SMS in proper manner, more than 200 archived data of patients have been considered. Those data are stored in JSON format. Various sizes of JSON strings have been taken for a simulation study and the results have been analyzed. Table 4.5 describes some of the results.

Figure 4.21 shows how the compression ratio changes with the length of characters. Figure 4.22 shows the relation between the character length before and after compression.

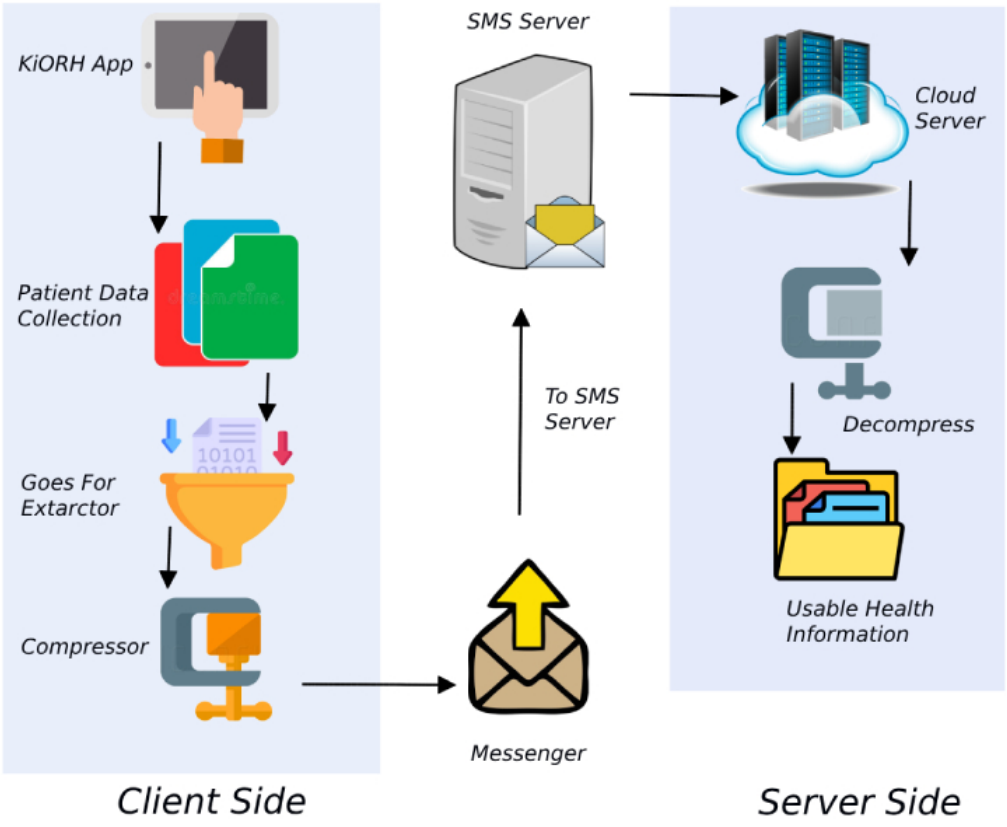

Figure 4.20: Flow of SMS sending

Table 4.5: Before and after data compression

Character length before compression	Character length after compression	Compression ratio
15	10	67%
19	12	63%
21	14	67%
42	23	55%
46	26	57%
55	33	60%
89	48	54%
106	49	46%
128	58	45%
179	83	46%
241	98	41%
279	108	39%
321	137	43%
398	151	38%
441	167	38%
478	179	37%

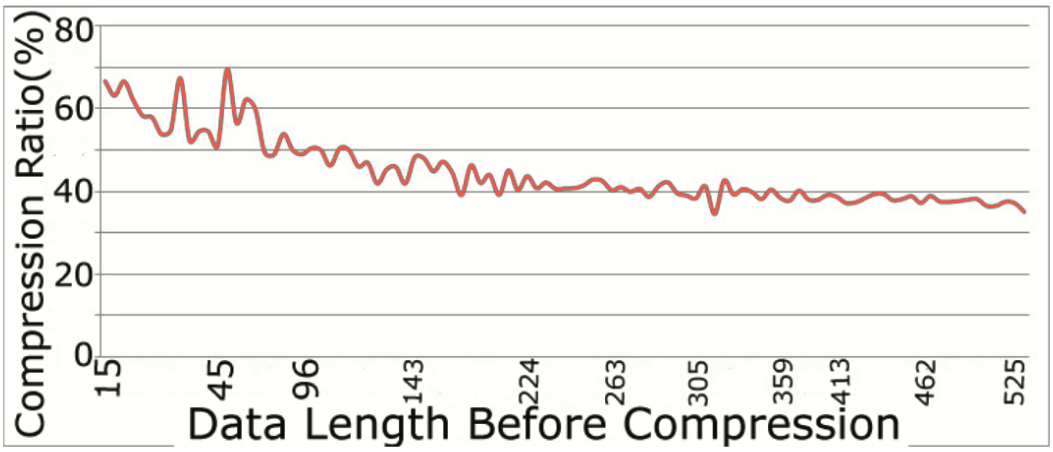

Figure 4.21: Comparison of compression ratio

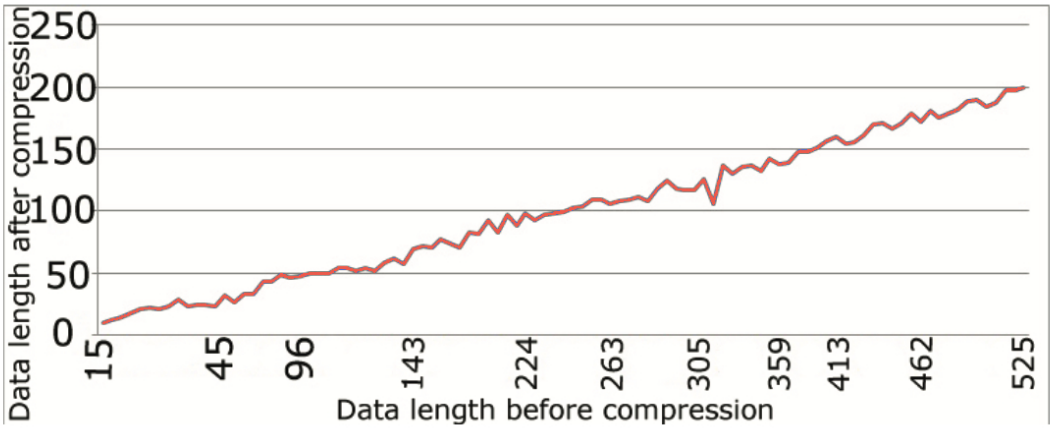

Figure 4.22: Character length after and before compression

4.9.4 Experimental observation and message length fixation

From the above results, it is observed that the compression ratio is maximum when the character length is between 50 to 70, and after compression, the character length becomes 30 to 40. But, as the maximum number of characters for one single SMS is 160, therefore the application makes its best fit and attempts to make the compressed character length 150 to 160. The application extracts the patients' vitals in such a way that the length of the JSON string characters of the symptoms, demographic information , and clinical examination should be within 450 characters combined as 450 or fewer characters can be compressed to 160 or fewer characters.

4.10 Summary

This chapter focuses on an alternative communication technique through mobile ad-hoc networks and cellular network using SMS when Internet connectivity is not properly available. In this chapter, it is also shown that the two proposed techniques can be used in the health-care domain. Based on the study of the technical challenges of video streaming over MANETs, new techniques and algorithms have been developed to improve the quality of video streaming. Also, a testbed has been created for transferring high-quality video in an ad-hoc network with the help of BATMAN-adv protocol. It has been demonstrated that the video streaming application works better for video surveillance (streaming and storing) where less number of pixel changes occur in each frame.

Therefore, for less dynamic applications, such as for rescue operation in disaster situation or for interaction between patient and doctor, the proposed approach will prove to be effective, because the changes in surroundings or video objects are not very fast. The approach is also general enough. Though it has been implemented on top of the BATMAN-adv protocol, it should work on other protocols at the data link and network layers. However, the approach will not be very effective in a highly dynamic environment or when the video scenes change very fast. There are other drawbacks as well to the use of unstable ad-hoc networks. It has been observed that the FPS drastically reduces after 4 to 5 hops. The ad-hoc mess network is built using BATMAN-Adv. So the path re-establishment time is much higher than usual, nearly around 2 to 3 seconds. Now if all the nodes are mobile, then some issues will occur as many video frames will be lost during the path re-establishment time.

On the other hand, this chapter also presents an alternative communication technique through SMS. The application KiORH captures patient symptoms, sends the data to doctors via the Internet or using SMS over cellular network when the Internet connectivity is low, and uses LZ77 and LZ78 compression algorithms to compress health data. The compressed health data is sent using SMS, with the compression process that reduces data from over 3000 characters to around 160 characters, ensuring it can be transmitted efficiently in low-connectivity areas,

improving access to healthcare for rural patients. The solution has been successfully implemented in two rural health kiosks in West Bengal, India.

Along with transmission of data, storage and retrieval of data is an important aspect of health-care services. It has already been discussed that health data is Big Data for its variety, veracity and velocity. Thus, modern Big Data tools can be efficiently used for storage and retrieval of health data. In the next chapter, an overview of multiple Big Data tools is presented.

Data is the new oil. It's valuable, but if unrefined it's useless. The refinery is the idea.

— Clive Humby, British mathematician and data scientist

5

Big Data in health-care

Contents

5.1 Prelude

With the recent advancement in distributed processing, sensor networks, cloud computing and similar technologies, Big Data has gained importance and several Big Data applications can now be envisaged which could not be conceptualised earlier. The health-care system consists of a large volume of data which are usually generated from diverse sources such as physicians' case notes, hospital administration notes, discharge summaries, pharmacies, insurance companies, medical imaging data, laboratory data, sensor waste devices, genomes, social media as well as articles in medical journal. This data is in various forms, mostly the large data is unstructured. This huge amount of data needs to be stored and analysed every day. In order to manage, store and process millions of health data generated everyday, the industry needs scalable and flexible infrastructure. Gradually as technologists focus on storing processing and managing Big Data, several Big Data solutions have come up.

Big Data tools and NoSQL databases become the necessary tools that store millions of data and respond to given queries efficiently. Traditionally, relational databases are used for many applications, but they are not at all efficient for storing and processing Big Data. The vast volume, as well as the complexity of health-care data make it difficult for the data to be processed and analysed by the traditional relational database approaches and techniques. Consequently, technologies such as cloud computing, virtualisation are now gradually being used for processing data effectively and securely. The drawbacks of relational databases motivate developers to use the NoSQL databases and Big Data tools while creating the health applications.

In this chapter, various characteristics of Hadoop, HIVE, Cassandra, MongoDB, HBase and Neo4j are studied. In the next chapter the performance of some tools are compared to respond to different queries to store health data for a remote health framework. The present study helps to come up with a few new ideas that can

provide support for storing and retrieving health data of a particular application using a data model discussed in the Section 3.3.2.

Health data is stored in different formats according to each tool. The performance evaluation of these tools against a selected set of queries, using the same experimental setup has been performed. The results are reported in the subsequent chapters.

5.2 Impact of Big Data in health-care systems

In recent times, the health-care system has witnessed a rapid increase in the use of electronic health-care systems to improve the quality of patients' care. However, the huge increase in the volume of health-care data has made it difficult to be effectively processed by traditional data processing applications. To efficiently extract values from health-care data, it is needed to understand the importance of Big Data in health-care. This section describes the need for analysing Big Data in health-care with the recent tools and technologies.

1. **Evidence-based care:** The standard medical practice is shifting from relatively ad hoc and subjective decision-making towards evidence-based health-care systems. Evidence-based medicine is a system in which the treatment of patients depends largely on available scientific evidence. Big Data provides evidence-based care by aggregating data sets from different data sources. The pattern in the data will provide enough evidence for checking, diagnosing and treating patients. The small health data sets might not provide sufficient evidence to determine if statistical differences are present in the data sets.

2. **Cost of health-care:** In recent days, the major challenges the health-care industry is facing is the increase in health-care costs. However, a recent survey carried out by a health research Institute revealed that more readily available information can reduce the cost of health-care [36]. The survey also indicated that patients are likely to opt for non-traditional forms of health-care services. Another group of researchers estimated that analysing the data with

innovative tools and techniques would result in a saving of $300 billion per year in the United States [94].

3. **Increased patients' participation:** Big Data ensures that patients have access to accurate and up-to-date information which enables the person to understand their choices, make decisions concerning their care as well and improve their lifestyle to avoid chronic diseases.

4. **Improvement in public health surveillance:** Big Data ensures the early detection and identification or diagnosis of diseases. This guarantees that the right decision on the treatment of a particular disease was taken in an effective and timely manner to stop and reduce patient morbidity and mortality.

5. **Increased communication between doctors and patients:** Big Data enhances effective communication between the health-care provider and the patient. Patients with similar health issues as well as health-care providers with similar specialities across the globe, can exchange ideas on the prevention and cure of the diseases.

6. **Early detection and prevention:** Big Data can be used to easily identify patterns and irregularities that indicate the presence of threats and abnormalities in health-care.

7. **Quality of care improvement:** Big Data improves the quality of care by delivering relevant and up-to-date results after making the decision by Big Data analytics.

5.3 Limitations of Big Data in health-care

There is no doubt that Big Data has a positive impact on the health-care system. The major challenge confronting Big Data in health-care is not the lack of data, but the lack of information to support decision-making, planning and strategy [50]. Health data needs to be validated, processed, and integrated to derive meaningful insights. This section describes the challenges of Big Data in health-care.

1. **Resistance to Change:** The health-care system tends to be late in adopting technology, unlike other sectors like the banking sector or the oil and gas industry. This is because of the inadequate administrative support for the information technology related practice changes, lack of trust, legal issues, as well as the lack of necessity and appropriate skills for the operation of ICT tools [56]. Thus, the paper-based system is still deployed in the health-care system and there is no support for the integration of data from diverse sources in the health-care system.

2. **Fragmentation of Data:** The challenges of Big Data in health-care are compounded by fragmentation and dispersion of data which is stored in proprietary heterogeneous systems across health-care organizations. health-care data are also stored in legacy systems which have limited inter-operability capabilities. Thus, it is difficult to integrate health data to form Big Data because there are different schemes, formats, metadata and standards underlying the data.

3. **Ethical challenges:** Ethical challenges, such as data privacy, confidentiality, control of health information access, the commercialisation of de-identified patients' information, and ownership and governance of health information hinder the effective exchange of information between the patient and health-care providers. The integration of health-care data from diverse sources becomes a challenge. Hence the access to a detailed and complete picture of a patient during care on time becomes a problem.

4. **Inadequate Standard:** Standards are agreed-upon specifications that allow disparate systems, tools, technologies and platforms to work together. However, health-care institutes do not maintain a single standard. For instance, the titles and other details of case reports, drugs, diseases and examinations vary in different hospitals [56]. So, the lack of a common standard in the health-care system makes it difficult to ensure portability among the heterogeneous systems.

5. **Security and privacy:** One of the major challenges for the integration of diverse sources of health-care information into Big Data is security and privacy. health-care information is harmed by security threats, such as improper disclosure of health information of patients, unauthorised use of demographic information and unauthorised destruction of health data. So, health-care providers become discouraged from sharing health information using electronic health-care systems.

5.4 Big Data in health-care: literature survey

The health-care industry is an indispensable entity in the real world where large volumes of data are accumulated from smart health-care devices, medical equipment, and digital medical appliances with the help of medical technology, information systems electronic medical records. It is very difficult to analyze large valume of health data using traditional approaches. So it is clear that the use of Big Data tools and techniques will be necessary.

A major challenge for health-care researchers is the storage and processing of various data types, including unstructured, structured, and semi-structured data. It is essential to do research for handling large data volumes, addressing complex system modeling, and resolving issues related to data source derivation. Early decision-making based health-care systems need to address the cost and quality of care and reduce waste and errors. Recently some survey have been made for the selection of proper Big Data tools based on their features. Some are carried out for machine learning applications, some are carried out for data analytics, while others are carried out for storage purposes

Lambey et al. [73] emphasized on the health-care recommendation system. The authors made a survey based on collaborative filtering techniques like content-based filtering, collaborative filtering, and hybrid filtering. The authors divide the health recommendation system mainly into four categories. One is intelligence-based (built with machine learning), second is Privacy Preserving Collaborative Filtering for Health Recommendations (PPCF) which overcomes different issues like

legal, financial, privacy and hybrid model-based, third one is Hybrid Recommender System and fourth one is Smart Recommender System which used multiple learning methods with a hybrid approach

Rahul et al. [64] give a brief overview of Big Data analytics in health-care. The authors first pointed out the challenges like processing unstructured health data, lack of skilful workers, slow migration to new technologies within the organizations, less adaptation of the latest tools etc. Then the authors highlighted the various data sources like mobile wearables, hospitals EHRs, health-care, medical insurance claims, pharmacy data, patient catalogues, government policies, and social media in the health-care domain. The authors gave an overview of the tools used for Big Data storage and processing along with the algorithms used in Big Data processing.

In another research paper [23], authors proposed a Big Data Intelligent Medical Platform (IMP) to efficiently acquire, synchronize and manage medical Big Data from various sources. IMP provides patient-centric health-care applications and services. The health data comes from the various sources and is stored in the Hadoop private cloud. The data provides continuous responses in real-time for data analytics and visualization with the help of Apache Hive.

Goyal et al., in their survey paper [47] elaborate on the steps of data analytics in the health-care domain. The authors also discuss the deep learning-based greedy approach and subspace clustering technique for the classification and clustering of health data.

In another research work [78], author Madyatmadja et al. gave an idea about Big Data analytics in health-care for understanding formats and trends in medical record data by discovering decision trees. The authors used Cardiovascular disease data sets which were collected at the moment of medical examinations in this research work. The dataset has 70,000 rows of patient data with 12 attributes of information about the patients and the results of medical examinations.

Multiple studies are going on which are related to the Big Data for the health-care domain, and researchers are trying to find the proper tools for storage and analysis. Hadoop, HDFS [97], CDH [53], MongoDB [80], Apache Spark [24], Apache Solr [48],

Pentaho Data Integration tools [104],Alteryx Designer [109], Data Meer [109], Google Big Query [109] etc are the tools used to store, process and analyze the Big Data. On the othe hand, Support Vector Machine [57], Neural Network [85], Logistic Regression [128], Linear Regression [128], Nearest Neighbor, Decision Tree, Naive Bayes are the algorithms used in health-care analysis. In short, researchers are trying to find the proper tools for efficient data storage and retrieval for some medical queries commonly asked by medical practitioners.

In the subsequent sections, brief description of the Big Data tools is given.

5.5 Tools and technologies for managing Big Data

.

With the advancement of information and communication technology, most especially the Internet of Things (IoT), health-care data has escalated from exabyte to petabyte and it is gradually approaching zettabyte and yottabyte [38]. With the astronomical rate of data growth, the high speed at which the data is generated and the complexity of health-care data, it becomes difficult for traditional data architectures to efficiently handle data sets in health-care. During the past two decades, many tools and technologies evolved for efficient management of Big Data. This section describes some of the sophisticated tools, technologies, applications, and platforms for managing and analysing Big Data in health-care. These tools are generally referred to as Big Data analytics. Some of the Big Data analytics tools are listed below.

5.5.1 Google Big Query

Google Big Query uses Google's cloud infrastructure to store and query massive data sets in a few seconds. Data is protected with multiple layers of security in Google Big Query.

5.5.2 MapReduce

MapReduce is a programming model for processing and generating large data sets in a distributed fashion [89]. This programming model works using two main thoughts - one is the map and the other is reduce. Users specify a map functionality that executes on the large data set and generates an intermediate key-value pair. The reduce function merges all intermediate values with the same intermediate keys [105]. Conceptually the MapReduce form is described as follows -

An array of key-value pairs $\{(k_1, v_1), (k_2, v_2), \ldots, (k_n, v_n)\}$ is considered.

The Map function processes a key-value pair of the array (k_i, v_i) and produces an intermediate key-value pair array (k'_j, v'_j), The mathematical expression is given

$$\text{Map}(k_i, v_i) \rightarrow \{(k'_1, v'_1), (k'_2, v'_2), \ldots, (k'_m, v'_m)\}$$

The intermediate key-value pairs are then shuffled and sorted.

$$\{(k'_1, \{v'_{1,1}, v'_{1,2}, \ldots, v'_{1,p}\}), (k'_2, \{v'_{2,1}, v'_{2,2}, \ldots, v'_{2,q}\}), \ldots\}$$

The Reduce operation takes each key and a list of associated values, processing them to produce a final output key-value pair:

$$\text{Reduce}(k'_j, \{v'_{j,1}, v'_{j,2}, \ldots, v'_{j,p}\}) \rightarrow (k''_j, v''_j)$$

The programs are written in such a functional pattern that are automatically parallelized and executed on a large cluster of commodity machines. The system takes responsibility for various things like partitioning the data, scheduling the program execution across all distributed machines, handling failures, managing internal processes and internal machine communications etc at runtime.

5.5.3 JAQL

JAQL is a functional and declarative query language designed to process large data sets. JAQL uses a MapReduce task to convert high-level queries into low-level queries to facilitate parallel processing. [119]

5.5.4 Hadoop

Hadoop is an open-source software framework that processes large amount of data across massively parallel clusters of servers. It is a non-relational database management system. The key component of Hadoop is the Hadoop Distributed File System (HDFS) which manages data across different servers. HDFS stores data of diverse types and structures.

With the recent advancement of Big Data solutions, scientists have focused on applying the Big Data concept following the MapReduce paradigm. The Big Data storage model based on Map-Reduce text categorization has been used for forecasting data in the next period in a particular region [96]. The two approaches are compared, i.e. MapReduce and GPU-Reduce to calculate the performance measurement for searching index files in database query processing. Various studies are made to check the performance of the NoSQL database system using the Map-Reduce programming model with single and multiple nodes. It is shown that NoSQL with MapReduce programming model provides better performance gain compared to relational database systems [121].

The Hadoop framework [26] handles distributed file systems with the help of HDFS (Hadoop file system). HDFS runs on master/client architecture. An HDFS cluster consists of a single Namenode or Master Node and several Damenodes. A Damenode stores the data files in the cluster, however, the Master Node does not store any data file. The Master Node manages the file system namespace and regulates access to files which are stored in the Damenodes. Besides, there is a secondary Namenode, usually, one per cluster, which manages storage attached to the nodes and helps the Master Node. If for some reason, the Master Node is down, the secondary Namenode can take the responsibility of handling the task of the Master Node. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of Damenodes. By default, HDFS maintains the data block size as 128MB. The Namenode / Master Node executes file system namespace operations, like opening, closing, and renaming files and directories. It also determines the mapping of blocks

onto Damenodes. The Damenodes are responsible for serving read and write requests, block creation, deletion, and replication upon instruction from the Namenode.

At the time of data processing in HDFS, the Hadoop framework sends the user-given analytics code to all of its nodes to run the user-given queries. Sometimes it may happen that the given query runs on such a node where no data is returned as output of the query execution. Therefore, processing on such nodes increases the overall data retrieval time.

5.5.5 NoSQL

In a relational databases, data is stored in rows and columns and can be accessed through structured query language SQL. In the cases of non-relational databases data is stored and retrieved through key-value pairs which provide links to where files are stored on disks. A non-relational database does not have a strong mechanism that ensures proper security [119]. Nowadays the above issue has been addressed and NoSQL databases are widely adopted in health-care applications.

- **Basic Principle of NoSQL:** NoSQL databases are especially useful for analysing a massive amount of unstructured and semi-structured data. NoSQL databases use completely different data structures for faster data processing. NoSQL depends upon three main theorems, which are described below [115].
  - **CAP Theorem:** CAP theorem [51] is also known as Brewer's theorem. CAP is known as C — Consistency, A — Availability, and P — Partition Tolerance. In 2000, Professor Eric Brewer put forward the famous CAP theorem. The core idea of CAP theorem is that a distributed system cannot meet the three distinct needs (as mentioned above) simultaneously, but can only meet two. According to the CAP theorem and different concerns of the NoSQL databases, there are three orientations to design a system, they are CA, AP, and CP. Availability is very important factor. So when designing the system, there are only CA orientation and AP orientation left. For the websites, availability and partition tolerance are

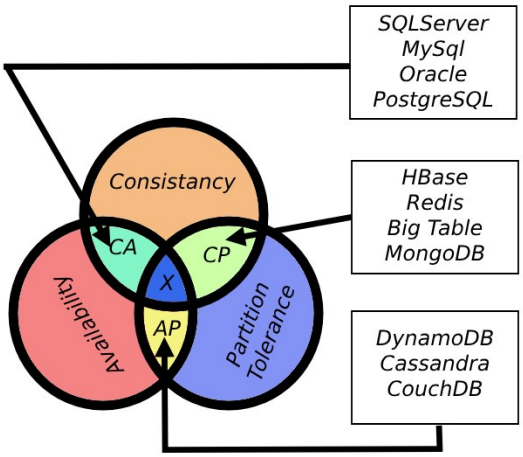

Figure 5.1: CAP Theorem

more important than consistency. It is sufficient when the system meets the eventual consistency (described below).

– **Base Theorem:** NoSQL database is typically said to be BASE compliant. BASE is the acronym for, BA — Basically Available (which means the availability of data is ensured by spreading data across many storage systems with a high degree of replication), S — Soft state (means in a period, the state of the system could be non-synchronous), E-Eventually Consistent (which means the system will eventually become consistent once it stops receiving inputs). The BASE model is completely different from the ACID model, which is well-known for relational databases.

– **Eventual Consistency Theorem:** Eventual consistency is one of the consistency models used in the domain of parallel programming. It means that given a sufficiently long period over which no changes are sent, all updates can be expected to propagate eventually through the system and all the replicas will be consistent. Consistency could be divided into two sides,

  * **The Client-side Consistency:** Client-side consistency has to do with how and when observers see updates made to a data object in the storage systems. There are two types.

- **Strong consistency:** After the update completes, any subsequent access will return the updated value.
- **Weak consistency:** The system does not guarantee that subsequent accesses will return the updated value. Several conditions need to be met before the value will be returned. The period between the update and the moment when it is guaranteed that any observer will always see the updated value is called the inconsistency window.

* **The Server-side Consistency:** On the server side, the consistency level could be modified because it is flexible. This flexibility affects how updates propagate through the system and the guarantees the system can provide regarding these updates. Consider the following notations:
  - N = the number of nodes that store replicas of the data
  - W = the number of replicas that need to acknowledge the receipt of the update for the update to be considered complete
  - R = the number of replicas that are contacted when a data object is accessed through a read operation

  If $W + R > N$, then the write set and the read set always overlap, and one can guarantee strong consistency [39]. However, when $W + R \leq N$, weak/eventual consistency could arise. It is possible that the read and write sets will not overlap.

5.6 Handling Big Data

The world of data is constantly evolving, and the amount of generated information is growing at an exponential rate. This phenomenon is known as Big Data growth, and it has a profound impact on nearly every aspect of our lives. 90% of the data in the world today was created in the last few years. The growth of the data increases while the storage media cost decreases.

It should be understood that the best use of technology for solving the problem is by keeping in mind some historical context, the evolution of systems and technology.

5.6.1 Big health data storage

As a centralized storage paradigm, cloud storage suffers from system failures, power outages, or even malicious attacks, which result from data loss in the cloud. Decentralized storage offers superior resilience in many cases because Big Data is redundantly distributed to many different storage providers. Hence, over the years, Big Data storage patterns have changed drastically. Every tool available in the market uses a separate storage process and pattern. Most of the tools use decentralized distributed storage for storing a large amount of data. From the breadth to the depth, the rapid reading and writing of massive data need some extra attention, because traditional data storage models and data compatibility relationships cannot handle all. Author Zhou et. al. [124] in their research paper developed a relational data extension model for cloud computing by processing the data features. The authors optimized the Big Data storage by creating a high-level portal system using the underlying database information. When the amount of data is relatively large, it is necessary to optimize and store the temporary data in the database cache system by clustering, to realize the design goal of converting and compatible a large amount of data. For real-time processing, fast scheduling and implementation of data tasks design is implemented. After the task data is collected, the required data are transmitted and stored. As the storage is not centralised, continuous integrity checking is needed. An efficient continuous Big Data integrity checking (CBDIC) approach for decentralized storage has been proposed by author Haiyang et. al. [120]. by designing a data-time sampling strategy that randomly checks the integrity of multiple files at each time slot with high checking probability.

The Big Data tools effectively use relevant storage patterns, such as data tables, data warehouses and datamarts, to make the best use of the data.

Table 5.1 shows the comparison of the storage patterns.

Properties	Data Lake	Data warehouse	Data Mart
Usage	• Big Data analysis • Machine Learning • Data insights	• Business Intelligence • Reporting • Data Analytics • Data visualization	• First lookups for summary Data
Data source	• Internal systems • External systems	• Data lakes • Relational databases • Aggregated	• Data warehouses
Data source	• Unstructured • Semi-structured	• Transferred	• Summarized

Table 5.1: Comparison of Big Data patterns

To increase the effieciency of Big Data storage, most of tools use the data compression method. By compressing the data, the size of the data is reduced. But there is a problem. It takes a long time to compress and decompress.

5.6.2 Retrieval of Big Data

Health Data Retrieval is the most crucial part of Big Data analytics. Each tool uses its own method for faster retrieval. Hadoop and other Hadoop-dependent tools use HDFS as a backbone for their data retrieval process. Cassandra and MongoDB use their indexing scheme for faster data retrieval.

In this thesis an experimental suudy is presented involving a few Big Data tools. In the next Section the Big Data tools which are used for experiments in this thesis are discussed in details.

5.7 Overview of the tools

Overview of some existing Big Data solutions and their advantages and disadvantages are summarised in this section.

5.7.1 Hadoop

The origins of Hadoop can be traced back to Google's research papers on the Google File System (GFS) and MapReduce, published in the early 2000s. These papers presented distributed storage and processing, which laid the groundwork for large-scale data processing, leading Doug Cutting and Mike Cafarella to create the open-source web search engine "NUTCH," later transformed into a project at Yahoo in 2006 that aimed to improve web search speeds by distributing data and computations across multiple computers.

Hadoop was initially a subproject of the Apache Nutch web search engine project. It became a top-level Apache project in 2008, gaining recognition and support from the open-source community. Yahoo played a crucial role in Hadoop's development by adopting it for their search engine and becoming a major contributor to the project. This further enhanced the credibility and robustness of the Hadoop framework.

The Apache Hadoop was established in the year 2008 and formed by Doug Cutting and Mike Cafarella at Yahoo and the University of Michigan respectively [69]. In Hadoop, Hadoop Distributed File System (HDFS) is responsible for data storage services and the data is processed using the MapReduce framework. In an HDFS environment, a Name Node behaves as a Master and there can be one or more Datanodes which behave as client nodes. Job Tracker and Task Tracker are the two services that place the MapReduce jobs onto the nodes. Job Tracker is the Master in MapReduce and Task Tracker is the client in MapReduce framework.

Some important features of Hadoop are given below.

- **Massive Storage Capacity:** The Hadoop framework can store huge amount of data by breaking the data into blogs and storing it on a cluster of low-cost, commodity hardware systems.

- **Parallel Processing:** Hadoop employs a parallel processing model that allows it to process data in parallel across multiple nodes. This accelerates data processing and analysis, particularly beneficial for tasks like batch processing and data-intensive operations.

- **Support for Multiple Programming Languages:** Multiple programming languages, including Java, Python, are supported. This flexibility enables organizations to use their preferred programming languages for developing applications on the Hadoop framework.

- **Proven in Industry:** Many large enterprises and tech giants have successfully implemented Hadoop for various Big Data use cases. Its proven track record in diverse industries, including finance, health-care, and e-commerce, has contributed to its popularity.

  Hadoop's low-level optimisation for analytic workload makes it a powerful platform on individual computers as well as clusters of machines.

- **Scalability:** Hadoop comes with a new way to store and analyse data. Hadoop is designed for horizontal scalability, allowing organizations to scale their data storage and processing capabilities seamlessly by adding more commodity hardware nodes to the cluster.

- **Cost effective:** The open source framework is free and uses commodity hardware to store large quantities of data. Organizations can leverage low-cost hardware components rather than investing in expensive specialised equipment.

- **Fault Tolerance:** Hadoop's distributed nature provides inherent fault tolerance. If a node in the cluster fails, the data and processing tasks are

automatically redistributed to other nodes, ensuring continuous operation without data loss.

- **Storage Flexibility:** Unlike traditional relational databases there is no need to pre-process data before storing it. And there is your flexibility to store unstructured data like text images and videos. Users can store as much as data they want and can decide how to use it later.

- **Data Security and Access Control:** Hadoop provides mechanisms for securing data within the cluster. It includes features such as authentication, authorisation, and encryption, ensuring that sensitive information is protected from unauthorised access.

- **Self-healing capabilities:** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail and it automatically stores multiple copies of all data.

As an open-source software, Hadoop has a large and active open-source community. This community support ensures continuous development, improvement, and troubleshooting, making it a reliable and well-maintained framework. While the Big Data landscape has evolved, and alternative technologies have emerged, Hadoop continues to play a foundational role in the field of Big Data.

5.7.2 Components of Hadoop

The core components of Hadoop include

- Hadoop Distributed File System (HDFS): A distributed file system designed to store and manage large files across multiple machines in a cluster. HDFS is optimized for high-throughput, streaming reads and writes of large datasets. Key features include:

- Large Storage: Capable of storing millions of files, each potentially tens of gigabytes to petabytes in size.

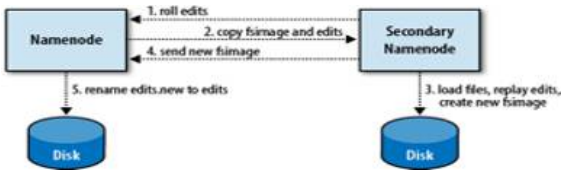

Figure 5.2: Hadoop namenodes structure

- Scalability: Uses inexpensive hardware and JBOD architecture to scale storage efficiently.
- Parallel Processing: Optimized for batch performance and high throughput, prioritizing large-scale data processing over quick access to small files.
- Fault-tolerance: Replicates data across nodes to handle machine or disk failures.

A standard Hadoop Distributed File System (HDFS) cluster consists of four key daemons: the NameNode, Secondary NameNode, Datanode, and Data Block [**Lam__2011**] [118].

The NameNode stores file-system metadata and manages the file-to-block mapping, maintaining a complete view of the file system. It is the central controller, running on a single node known as the MasterNode. The Secondary NameNode assists the NameNode with tasks such as checkpointing and editing logs, but it does not serve as a backup. Figure 5.2 shows the Hadoop namenode and secondary namenode structures.

Datanodes store the actual block data and can be deployed in large numbers across the cluster (known as Slave Nodes). The system stores large files by breaking them into blocks, typically 128 MB or 256 MB in size, which are distributed across Datanodes. Data replication ensures fault tolerance, so if a Datanode fails, its data can still be accessed from replicas stored on other Datanodes. Figure 5.3 shows the Datanode structure.

Example of block formation of 380MB and 500MB is shown in Figures 5.4 and 5.5

HDFS enables both read and write operations. For a read operation, a client contacts the NameNode to identify the block locations, after which it reads directly

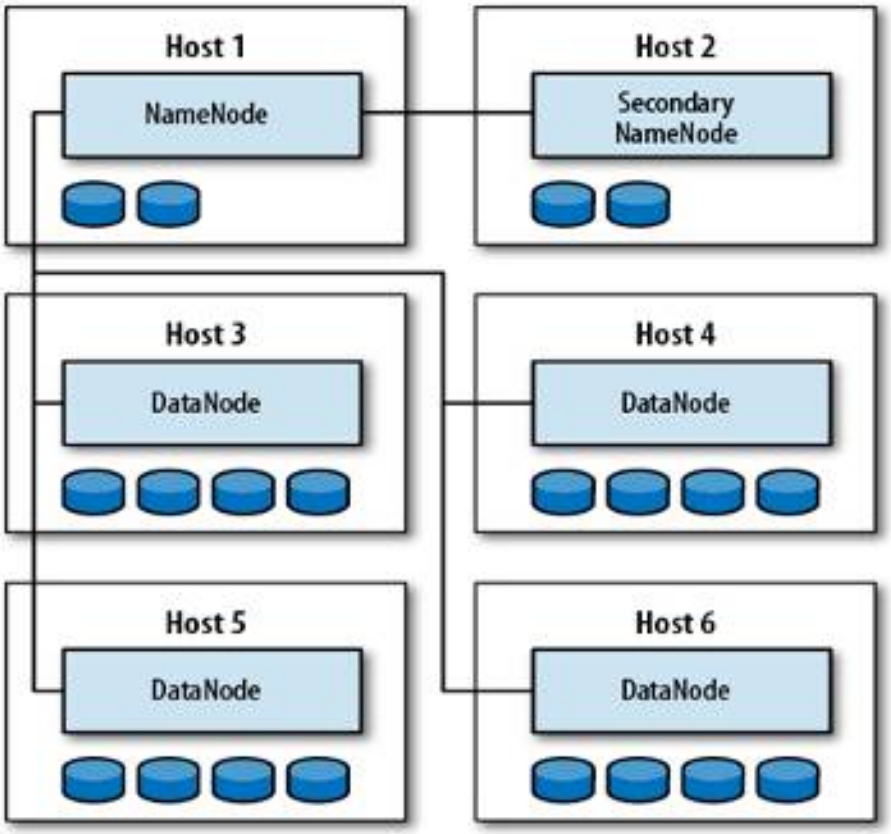

Figure 5.3: Hadoop Datanode structure

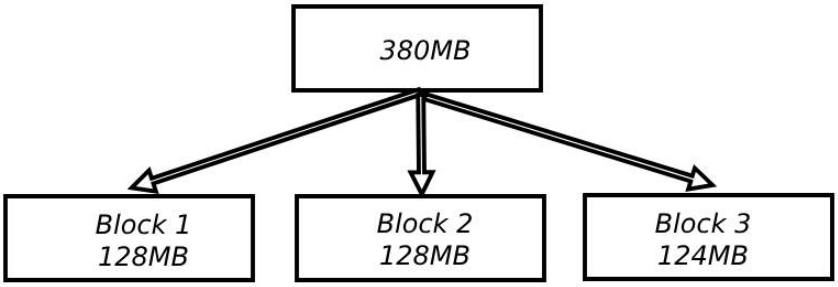

Figure 5.4: HDFS block formation of 380MB file

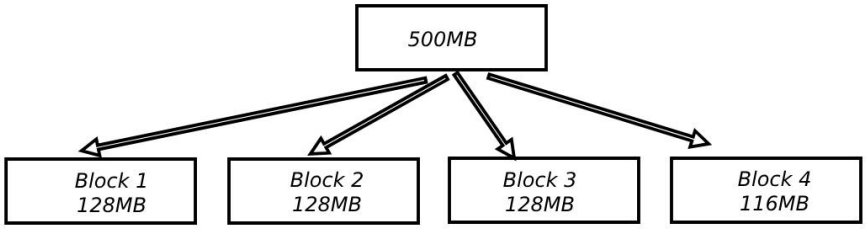

Figure 5.5: HDFS block formation of 500MB file

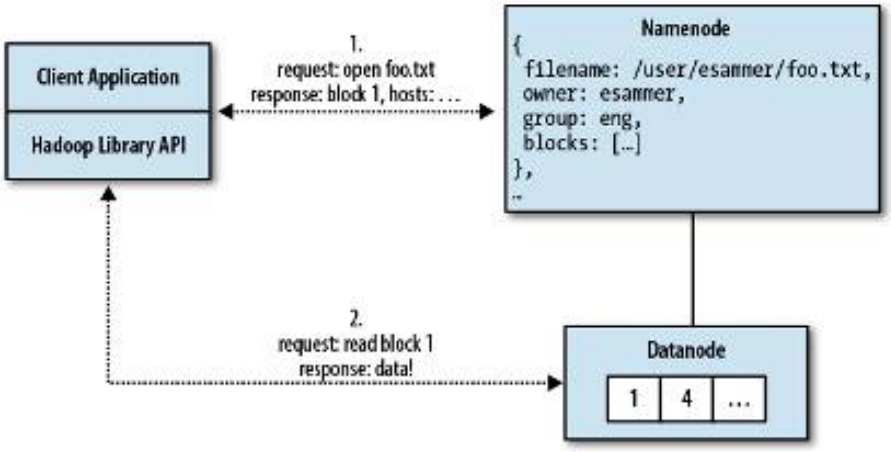

Figure 5.6: HDFS read operation

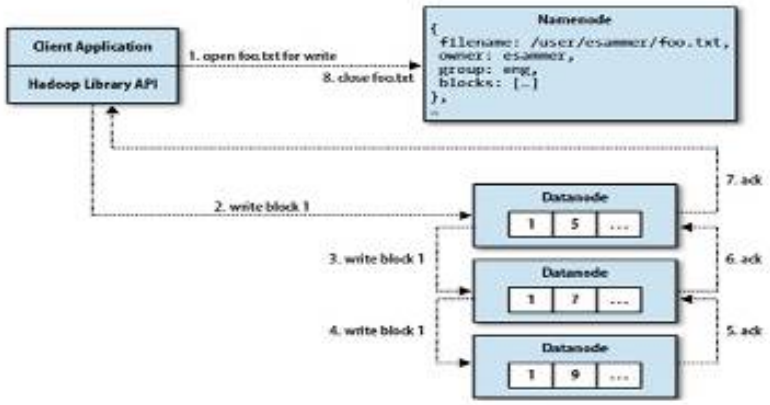

Figure 5.7: HDFS write operation

from the DataNodes. During a write operation, data is streamed through a replication pipeline across multiple DataNodes, ensuring fault tolerance. The pictorial representation of read and write operations are shown in Figure 5.7.

Hadoop also features MapReduce, a computational framework designed to work alongside HDFS. MapReduce leverages data locality, running tasks on the same machines where data is stored, thus minimizing network overhead. The two main MapReduce daemons are the JobTracker, which manages task scheduling and monitoring, and the TaskTracker, which runs the actual tasks on worker nodes. The map and reduce Phase are shown below in Figure 5.8 and Figure 5.9.

Hadoop has evolved with YARN (Yet Another Resource Negotiator), which separates resource management from processing, allowing various data processing

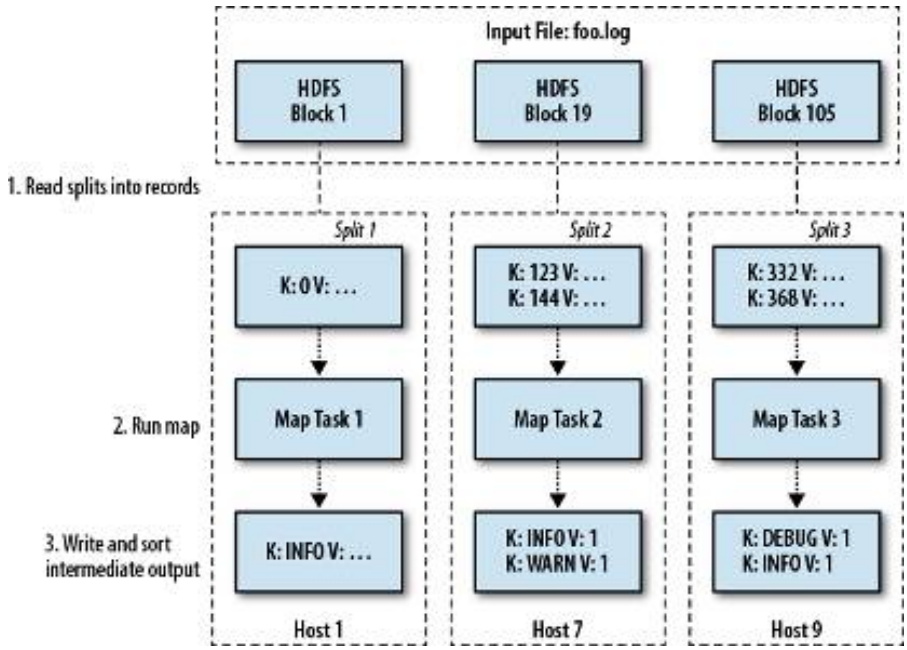

Figure 5.8: Hadoop MapReduce

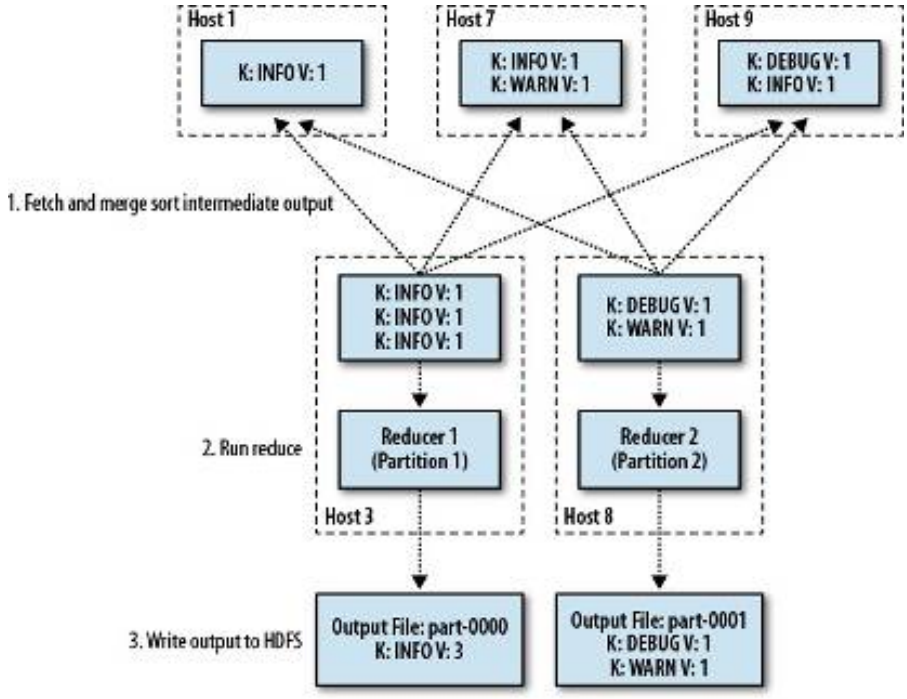

Figure 5.9: Hadoop reduce phase

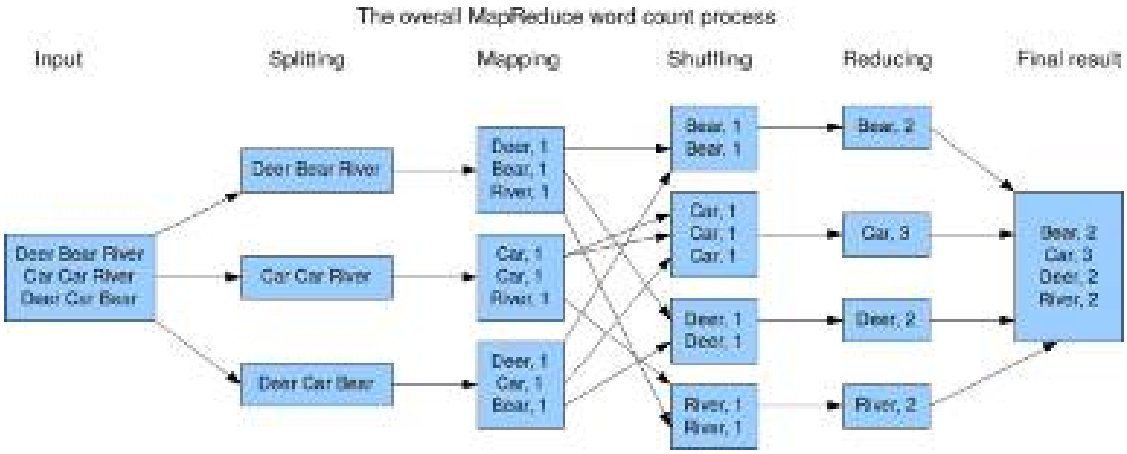

Figure 5.10: Simple word count program execution in Hadoop

engines like MapReduce and Apache Spark to share cluster resources efficiently. MapReduce 2 (MRv2) is a newer version of the MapReduce framework that utilizes YARN, enhancing scalability and flexibility by supporting multiple applications in a shared cluster environment.

MapReduce [118] is a distributed data processing framework that simplifies large-scale data processing by dividing jobs into a Map phase and a Reduce phase. It parallelizes tasks across slave nodes, monitors job progress, and recovers from failures.

The advantages and disadvantages [117] of Hadoop are listed in Table 5.2.

Advantages	Disadvantages
• Support for multiple languages • Compatible with other file systems • Ease of use • Scalable • Open source • High throughput • Low network traffic • Highly available • Fault-Tolerant • Good performance • Cost-effective • Varied data sources • Easy improper nodes/node decomposition	• Some issue with small files • Vulnerable by nature • Processing overhead • Iterative processing • Weak security

Table 5.2: Hadoop: Advantages and Disadvantages

5.7.3 Hive

Apache Hive is a data warehouse software project built on top of Hadoop for providing data analysis, summarization, and querying on a large set of data. Hive was initially developed by Facebook, but later it was used and developed by other companies like Netflix and the Financial Industry Regulatory Authority. It provides the original SQL framework on top of Hadoop and gives an abstraction layer on top of MapReduce to make it easier for analysts and enable data scientists to query data based on the Hadoop file system.

HIVE can take data from Hadoop, HDFS, and local file systems, and it can

write data to all of these. It offers efficient data aggregation, analysis, and ad-hoc querying on huge amounts of data set [32].

HIVE converts HQL(Hive Query Language) queries to a directed acyclic graph of MapReduce jobs [54]. These MapReduce jobs are executed in Hadoop.

Custom MapReduce scripts can be plugged into queries. Data can be queried in custom formats by using existing IO libraries. It has a metadata containing schemas and statistics that are used to explore data effectively. It supports primitive data types within tables and supports array, map, and nested structures. It works with text, binary, and column-oriented file formats [113].

The advantages and disadvantages of Hive are listed in Table 5.3.

Advantages	Disadvantages
• Keeps queries running fast • Takes very little time to write Hive query in comparison with MapReduce code • HQL is a declarative language like SQL • Provides the structure on an array of data formats • Multiple users can query the data with the help of HQL • Very easy to write query including joins in Hive • Simple to learn and use	• Useful only when the data is structured • Analytical operation cannot be done • Debugging code is difficult • Difficult to perform complicated operations • Does not support OLTP (Online Analytical Processing) • Only non-real or cold data is supported • HQL does not support the transaction processing

Table 5.3: Hive: advantages and disadvantages

5.7.4 HBase

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. It is based

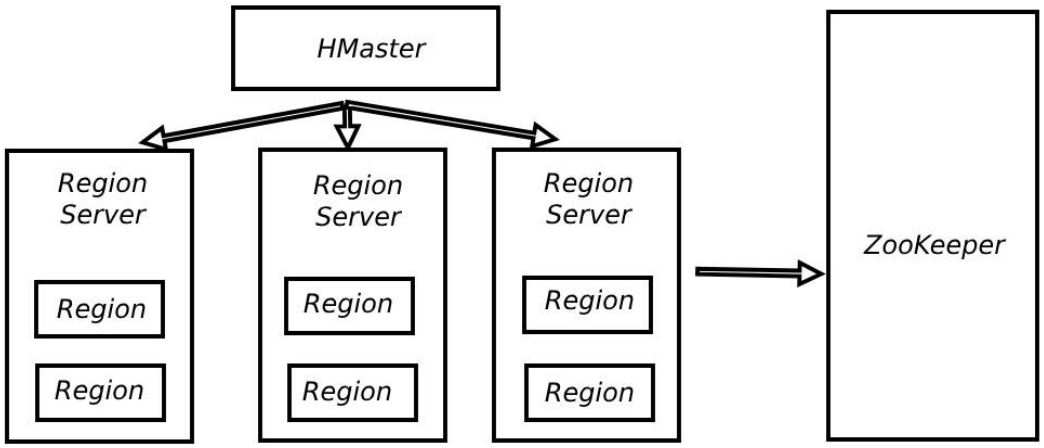

Figure 5.11: Basic HBase architecture

on a data model that is similar to Google's BigTable and is designed to provide quick random access to huge amount of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS). It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. HBase uses the MapReduce framework to process aggregated queries, which requires real-time computation [74].

Architecture of HBase The HBase architecture has three main components: HMaster, Region Server, and Zookeeper. The client contacts the HMaster. The HMaster allocates regions and load balancing. The Region Server provides services to read and write data by interacting with HDFS, and the Apache Zookeeper monitors the system. Figure 5.11 shows the HBase architectural diagram after including the client and HDFS. The HBase cluster typically consists of one masternode (HMaster) and many slave node as region servers.

- **HMaster:** The masters' responsibilities include coordinating the servers in the region, allocating regions during launch, re-assigning region for recovery, and load balancing. In addition, the master service is an interface for creating, deleting, and updating tables. HMaster acts as an interface for all metadata changes and takes responsibility for metadata operations.

- **Region Server:** The Region Server comprises all the machines of Hadoop cluster. Region Servers handle read, write, update, and delete requests from clients for one or more regions.

- **ZooKeeper:** Zookeeper is the distributed coordinator in HBase for region assignment and is used to recover from any crashes at a Region Server by reassigning the regions that were handled by the failed server to a functional Region Server. It keeps track of settings and synchronization of data across several machines.

The advantages and disadvantages of HBase are listed in Table 5.4.

Advantages	Disadvantages
• Schema-less, column-oriented • store denormalized Data • Supports automatic partitioning • Low latency operations • Easily integrates with Hadoop • Supports distributed storage • Network failover recovery • Linearly scalable data replication	• Traditional models, features not supported • Does not support SQL structure • Single point of failure • No built-in authentication • Does not have support for transaction • Joining and normalization are very difficult

Table 5.4: HBase: advantages and disadvantages

5.7.5 Cassandra

Apache Cassandra is an open-source distributed database management system designed to handle large amount of data across many commodity servers. Cassandra provides high availability with no single point of failure. It also offers robust support for clusters spanning multiple datacenters. The data model of Cassandra is a partitioned row store with tunable consistency. Rows are organized in a table;

the first component of the primary key of the table is the partition key; within a partition, rows are clustered by the remaining cloumns of the key. Other columns may be indexed separately from the primary key. This database works using a peer-to-peer architecture, where data will be replicated on multiple nodes in the cluster. Data stored in Cassandra is fetched using Cassandra Query Language (CQL).

The Cassandra database was created by Facebook, and its architecture is inspired by Amazon's DynamoDB database, while its data model is based on Google's BigTable. Currently, this database is maintained by the Apache Foundation [28]

Here are some main features of Cassandra

- **Decentralized:** Every node in the cluster has the same role, and there is no master. Data is distributed across the cluster so that each node contains different data, but every node can provide service for any request.
- **Supports replication and multi-datacentre replication:** Cassandra is designed as a distributed system, for the deployment of large numbers of nodes across multiple data centers. so replication strategies are configurable for failover and disaster recovery.
- **Scalability:** Read and write throughput, both increase linearly as new machines are added, with no downtime or interruption to applications.
- **Fault-tolerant:** Data is automatically replicated to multiple nodes for fault-tolerance. Failed nodes can be replaced with no downtime.
- **Tunable consistency:** Writes and reads offer a tunable level of consistency, all the way from 'writes never fail' to 'block for all replicas to be readable', with the quorum level in the middle.
- **MapReduce support:** Cassandra has Hadoop integration with MapReduce support.

The advantages and disadvantages of Cassandra are listed in Table 5.5.

Advantages	Disadvantages
• It is open-source • Peer-to-peer architecture, so no single point of failure • Easily scaled down or up • Fault-tolerant and has high availability • High-performance • Schema-free • Supports hybrid cloud environments, and can be deployed across many data centers	• Does not support ACID and relational data properties • Has latency issues • Data redundancy occurs because modeled around queries and not structure • Experience JVM memory management issues • No support for join or subquery • Fast writes, slower reading • Lacks official documentation

Table 5.5: Cassandra advantages and disadvantages

5.7.6 MongoDB

MondoDB is a cross-platform document-oriented NoSQL database. It eschews the traditional table-based relational database structure in favor of JSON like documents with dynamic schemas, which is very effective for processing Big Data. MongoDB delivers great performance, accessibility, and scalability. It is a document-oriented and cross-platform NoSQL database. All documents are clustered as groups based on their structure. MongoDB stores documents in BSON (Binary JSON) format. Through distributed key-value pairs, the design capability of MapReduce, and document-oriented features, it has attained enormous attention [27].

MongoDB offered two editions. community and enterprise. The enterprise edition includes additional corporate features such as LDAP, KERBEROS, auditing, and on-disk encryption. It uses the sharding method to distribute data across multiple servers to improve scalability and performance. Sharding method do the horizontal partitioning of data across multiple servers or nodes. It distributes large datasets into smaller, manageable chunks, called shards, which are stored

across different servers to balance the load, enhance performance, ensure high availability and fault tolerance.

There are three components of MongoDB.

- **Shard:** A shard is a single instance of a MongoDB database that holds a subset of the data in the shard cluster. Each shard can run on a separate server by independent MongoDB deployment.
- **mongod:** Mongod is the primary daemon process for the MongoDB system. It is responsible for managing data storage, retrieval, and other core database functions.
- **mongos:** Mongos (MongoDB Router) is a routing service that acts as an interface between the application and the sharded cluster. Applications connect to Mongos instances directly to individual Mongod instances.

The advantages and disadvantages of MongoDB are listed in Table 5.6.

Advantages	Disadvantages
• Support for in-Memory or WiredTiger [41] storage systems • Schema-less database architecture • Support for dynamic document queries • Scalable • Make index on any attribute • No need to map or convert application objects into database objects • No complex joins	• Does not support transactions • Requires more storage than other well-known databases • Does not automatically clean up its disk space • Not easy to combine two documents • Speed drops significantly if indexes are not correctly implemented or not in the correct order

Table 5.6: MondoDB: advantages and disadvantages

5.7.7 Neo4j

A database called a "graph database" prioritizes the links between the data as highly as the individual pieces of information. It represents and stores data using graph structures (node and edge). In graph databases, the record, object, or entity is represented by a node, and the relation between nodes is represented by an edge. Since graph databases are made to manage and store data in a graph style, they are effective at handling data with intricate relationships. There are numerous varieties of graph databases, each with unique advantages and applications. Neo4j is one of the best graph databases available, and strangely, it was created using Java. It also includes a language of its own, called Cypher, which is like declarative SQL but tailored to match graphs. Along with Java, it also supports several other well-known languages, including Python,.NET, JavaScript, and others. Neo4j is perfect for tasks like managing data centers and detecting fraud. Neo4j comes in two flavors: an open-source community edition with online backup and high availability extension and a pre-package licensed version with backup and extension. The advantages and disadvantages of Neo4j are listed in

Advantages	Disadvantages
• Graph Model • Native Graph Processing Capability • Powerful and expressive Cypher Query Language • High Performance on Connected Data • Flexible Schema • Supportive and active Community and Ecosystem	• Storage Overhead • Complexity for Simple Data Models • Learning Curve • Scalability Challenges • Resource Intensive

Table 5.7: Neo4j: advantages and disadvantages

5.8 Experimental study with Hadoop and Cassandra

An experimental study was conducted in [98] using Hadoop and Cassandra to analyze the query performance, storage, and retrieval of health data, including sensor observation data, in the health-care domain. A proposed health-care data model as shown in Figure 3.3, based on national and international EHR standards, was mapped onto these two Big Data solutions for executing a set of queries. The study observed that the mapping strategy is a critical factor in improving the performance of any Big Data solution. However, this strategy should be based on queries that are frequently used in the application domain. The experimental results demonstrated that Cassandra performs well for certain queries with effective use of partition keys. It was also observed that Hadoop performs better with large datasets compared to Cassandra. Hadoop is more focused on data processing, making the scheduling strategy crucial in Hadoop implementations. On the other hand, since data storage and distribution are the main goals in Cassandra, its implementation should prioritize data mapping strategies. It can be concluded that the choice of partition key is crucial for Cassandra's performance, and index keys are not always an optimal design decision.

However, as the data volume grows rapidly, the computation time increases accordingly. In cases where the node count increases, even with a small volume of data, the computation time increases due to the additional time required for data mapping and reduction.

The experiments were conducted with a dataset of over 800,000 records containing various types of semi-structured medical data to study the analysis time for various queries. The study revealed that processing time is relatively consistent for smaller datasets compared to larger ones. There is no significant time difference when data sizes are smaller or equivalent to the Hadoop data block size (64 MB). For smaller datasets, the map and reduce times are slightly longer than for larger datasets. Conversely, in the case of large datasets, analysis time increases gradually as the number of Hadoop data nodes increases exponentially (2^i).

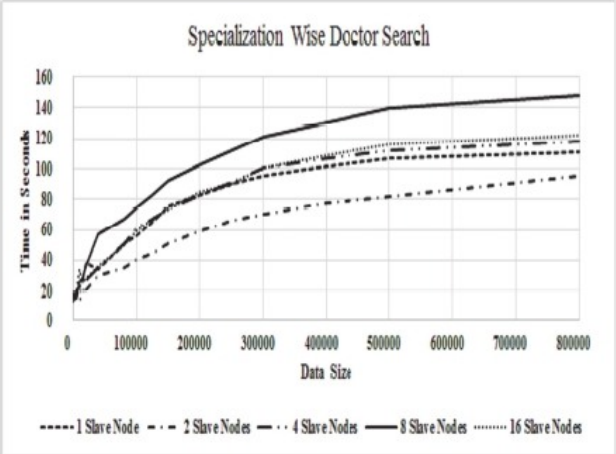

Figure 5.12: Query performance of specialization-wise doctor search in Hadoop

Some simple queries queries are processed by customized Hadoop Map-Reduce code.

The queries are given below.

1. Find all doctor's names and their details with specialization string 'cardiologist'.
2. Find all information of patients like demographic info, religion, food habit etc, for apatient.
3. Find all ECG reports of a patient.
4. Find history of 'Diabetes' of a patient.
5. Find sensor data of a patient for one hour.

Some key findings [98] are as follows:

1. Hadoop and Cassandra are designed for processing large amounts of data; statistical data analysis and query processing are not suitable for small datasets.

2. In a Big Data environment, the concept of non-relational data models, such as joining and complex queries, is limited. Therefore, redundant data needs to be stored in multiple locations.

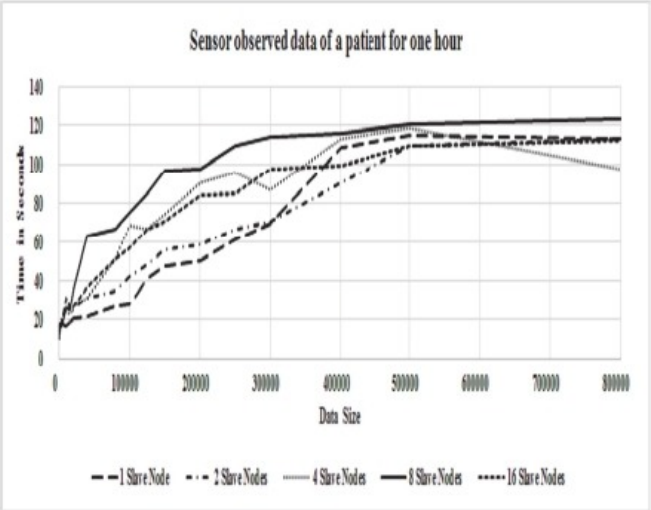

Figure 5.13: Query performance of Sensor observation data of a patient during an hour

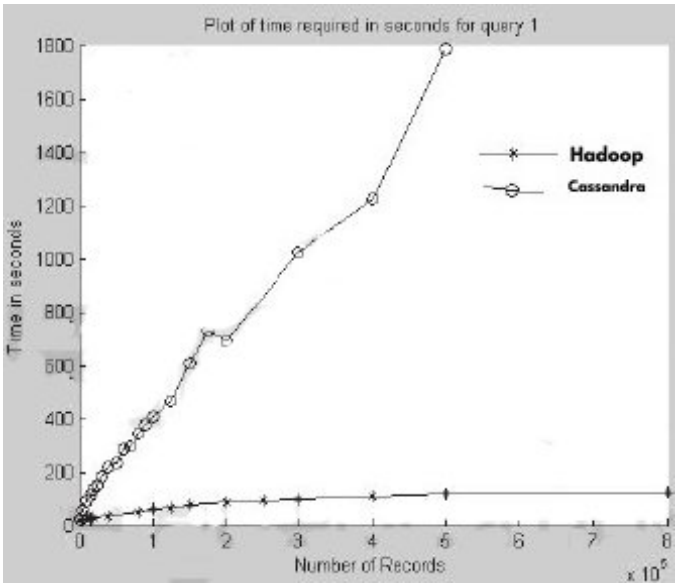

Figure 5.14: Query performance of specialization-wise doctor search in Cassandra and Hadoop

3. The non-relational data model for EHR must be nearly perfect according to the requirements. Otherwise, it will be challenging to change the model after implementation.

4. Data processing time gradually increases as the number of nodes changes with a fixed number of records. However, the reasons for the increased processing time are still unclear from the current experiment.

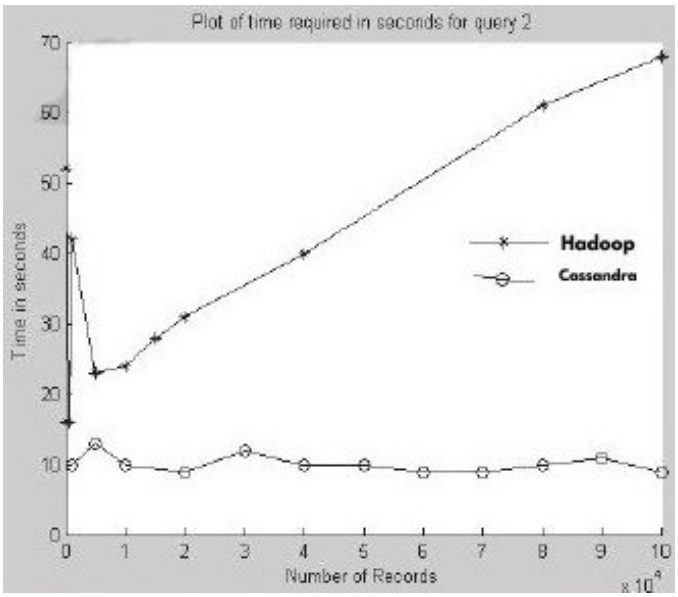

Figure 5.15: Query performance of patients demographic information in Cassandra and Hadoop

5.9 Summary

Big Data refers to large volumes of high-velocity, complex, and variable data that require specialized techniques and technology for capturing, storing, managing, distributing, and analyzing data. The primary goal of Big Data tools is to economically extract value from vast amounts of diverse data. In the health-care system, Big Data tools are essential for facilitating processing of heath data and seamless communication among health-care providers, hospitals, patients, and doctors. However, despite the numerous benefits of these tools in providing health-care services, several challenges hinder the effective adoption of these tools for health-care systems.

This chapter discussed several powerful tools that have emerged to address the challenges associated with large-scale data processing, each contributing unique functionalities. Hadoop, a distributed processing framework, allows efficient storage and fast processing of large volume of data across a cluster of computers. Hive simplifies data querying and analysis with its SQL-like interface, while HBase provides a NoSQL, column-family storage system on top of Hadoop, supporting real-time read and write operations. MongoDB, a NoSQL database, offers scalability and

flexibility by storing diverse data types in a document-oriented format. Cassandra, another NoSQL database, provides high availability and fault tolerance, making it suitable for distributed and decentralized architectures.

In the next Chapter 6, a novel framework is proposed for storage and management of Big Data. In Chapter 7, experiments and studies are conducted with other Big Data tools, and algorithms are developed.

Big Data is not about big numbers, it's about big knowledge.

— Viktor Mayer, Schonberger

6 Node Guided MapReduce

Contents

6.1 Prelude

In recent years, the emergence of the Internet and communication technology has led to an increasing rate of data generation. Everything around us is connected to the Internet, continuously producing data. Health-care is one of the largest data-producing sectors. To efficiently extract the required information from this vast amount of data, it is essential to store it in a distributed manner to ensure consistent retrieval times. This requires a massive storage cluster composed of tens of thousands of connected machines equipped with built-in analytics tools to store and retrieve data quickly and systematically.

Apache Hadoop is a well-known software library framework that allows for the distributed processing of large data sets across clusters of low-end computers using simple programming models. Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage. When the Hadoop Distributed File System (HDFS) ingests data, it breaks the raw data into separate blocks and distributes them to different nodes in a cluster for parallel processing. However, it does not allow determining which block is stored in which node. Consequently, when a query on a specific item or keyword is executed, Hadoop runs the query or analytics program across the entire cluster, which can increase the overall cost. In this chapter, a new framework is proposed to overcome the above challenges. In this framework, the data is distributed and replicated in a guided manner. Thus, MapReduce queries run faster on this framework.

6.2 Background

With the recent advancements in distributed processing, the health-care sector has become one of the largest data-producing sectors, generating vast amounts

of data daily. One of the major challenges is creating a large-scale distributed storage system that can efficiently store, process, and retrieve necessary health records. Since its inception, Hadoop MapReduce has become a popular tool for storing, managing, and processing massive amounts of data. Hadoop MapReduce offers a highly effective framework for handling Big Data. However, this framework distributes data across all its nodes, and during processing, the user-provided MapReduce queries are executed on all nodes. Consequently, some queries may be processed on nodes where no relevant data is stored, increasing the overall processing time. To address this issue, this chapter proposes a distributed storage platform and MapReduce model that overcomes the problems faced by existing systems. Based on the proposed model, a framework called Node-Guided MapReduce (NGMR) is developed using the Google MapReduce paradigm.

The NGMR framework guides the MapReduce process to run on a specific set of Datanodes among all nodes in the cluster.

6.3 Overview of proposed framework

The enormous size of Big Data files makes it impractical to store them in a single conventional storage system. These files need to be fragmented into smaller chunks and distributed among various storage nodes. Conventional distributed storage systems, typically composed of inexpensive hardware, form a cluster where each small chunk, tagged with a unique identifier, is stored on multiple nodes (clients). These chunks must be frequently searched, located, and processed based on business needs. The framework proposed in this chapter aims to optimize multiple dimensions, including spatial locality, unique identification generation for each fragment, and an efficient indexing mechanism for faster access and reduced query processing time. The primary idea is to store similar types of fragments (divided based on key attributes) within the same cluster whenever possible. This approach enhances spatial locality, reducing the major source of data analysis latency, which is the reading and writing of files.

The proposed framework is designed as a master-client system following a cluster-based architecture. At the top of the architecture is a Master node, with multiple Datanodes connected to it. The fragmented data is stored in these Datanodes. During processing, user queries are executed using the MapReduce framework, distributed among the Datanodes, and processed for each fragment. Upon successful execution, the results are accumulated and stored in the Master node for further processing or future use. The key features of the proposed framework are given in the following subsections.

6.3.1 Input and output types

The proposed system supports reading and processing input data in various formats. The data can be structured (e.g., SQL backup files, key-value pairs), unstructured (e.g., text files where each line is treated as a key-value pair), or semi-structured (e.g., CSV, XML, JSON, No-SQL database files, treated as line-based, tag-based, syntax-based, and format-based) [101]. Users can define the key for the input file and the associated values in a key-value pair format. They need to provide the data or a link to the cloud storage where the files are stored. The system will produce an output in the same file type as the input. Users can also define the desired output format.

6.3.2 Master-Client architecture

The Master node maintains several data structures for different tasks. For storage, the Master node keeps metadata information in JSON files for each file. The state information for MapReduce tasks, such as idle, in progress, or completed is stored. The Master node also stores information about the client nodes, including hostname, available space, total usable space, last heartbeat received time, and Datanode block sizes. The Master node serves as the conduit through which the location of intermediate block regions are communicated from map tasks to reduce tasks. For each completed map task, the Master node stores details of the intermediate file regions produced by the map task. This information is pushed to workers (Datanodes) that have in-progress reduce tasks. On the other hand, the worker

(Datanode) maintains various data structures, including metadata for all stored blocks for every file. The blocks are stored in text format. The details of metadata and blocks are described in Section 6.3.3.

6.3.3 Multilevel indexing

One of the most significant parts of the proposed system is the indexing of small blocks of large files or documents [98]. The indexing system addresses fault tolerance, parallelization, and distribution. The document size is assumed to be more than 10-20 terabytes, with the smallest block size being 64 MB. The indexing system accepts large files or documents, or the cloud link to the document provided by the user, as input. The system reads the documents horizontally or vertically, as needed or directed by the user.

In the proposed solution, it is needed to maintain two levels of indexing: one at the Master node level and another at the Datanode level. At the Master node level, indexing stores details of files or documents, such as names, respective cluster IDs, and references to child-level indexing. Child-level indexing, at the Datanode, stores information about all blocks, sizes, locations, and root file names. The raw indexing file structure at the Master Node is shown below.

$$DocumentName =< FileName.*^{+} >$$
$$KeyID =< DistributionAttributeKey^{+} >$$
$$Distribution =< KeyID^{+} >$$
$$DistributionAttributeKey =< DocumentContent >$$
$$ClusterID =< ClusterIdentifierIDs^{+} >$$
$$ClusterIndexRef =< ClusterID^{+}, ClusterFileName^{+} >$$

Similarly, the raw structure of the Datanode indexing file is as follows,

$$DocumentName =< ClusterIndexRef^{+} >$$
$$Distribution =< blockID^{+}, ClusterID^{+}, KeyID^{+}, FragmentID^{+} >$$

Structure of the both meta data are shown in Figures 6.1 and 6.2

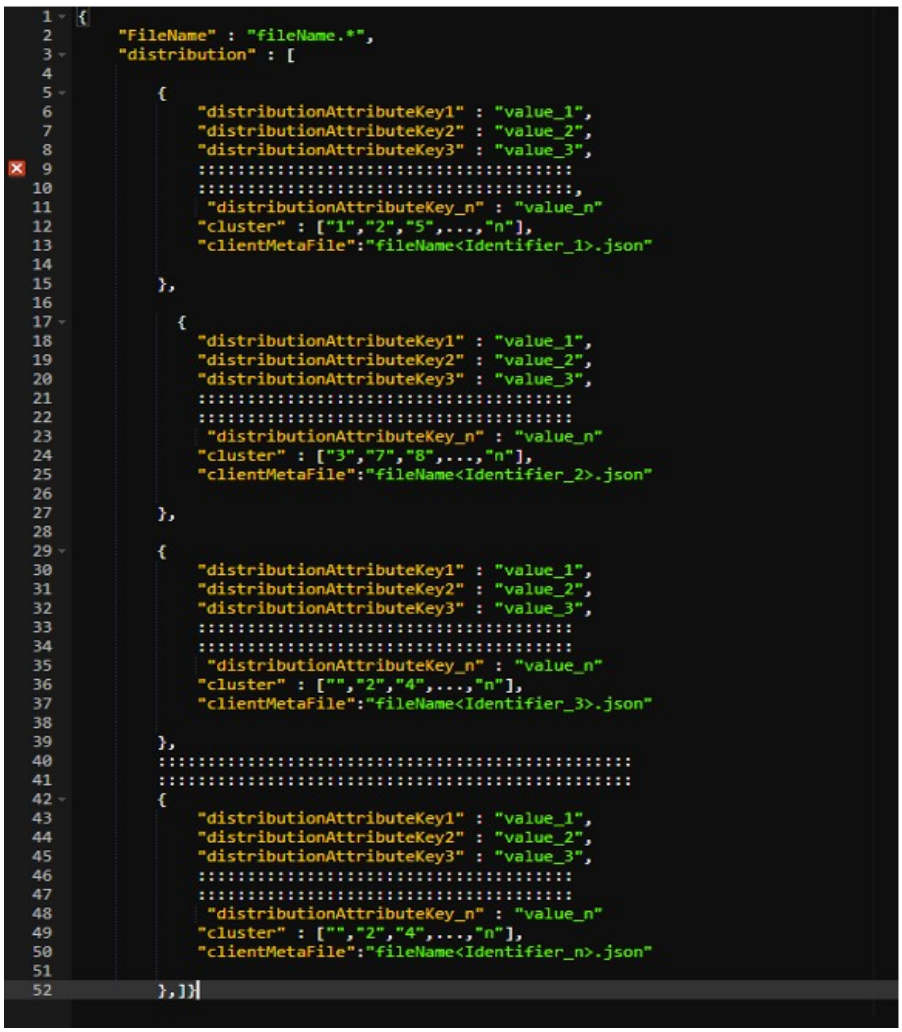

Figure 6.1: Master meta data structure

Figure 6.2: Datanode meta data structure

6.3.4 MapReduce framework

MapReduce is particularly well-suited for processing and analyzing vast amount of data across distributed computing environments. It provides fault tolerance and scalability by distributing the data and computation across a cluster of machines. Conceptually, the MapReduce framework can be described as follows:

Programs are written in a functional pattern that is automatically parallelized and executed on a large cluster of commodity machines. The system takes responsibility for various tasks, such as partitioning the data, scheduling the program execution across all distributed machines, handling failures, and managing inter-process and inter-machine communication in real-time. The MapReduce framework is discussed in details in Section 5.5.2.

6.4 Phases of the NGMR model

In this section, the four phases of the proposed NGMR scheme is discussed :

6.4.1 Fragmentation and replication phase

Generally, distributed storage and processing frameworks require more read operations than write operations. The proposed scheme works efficiently in such scenarios, but also performs well when both read and write operations occur frequently. A set of keys based on the attributes of the Big Data file is provided as input by the end user to fragment and distribute the Big Data files. The keys can be individual attributes, a collection of multiple attributes, or a combination of both. If the length of the combination of keys is greater than three, this length is used as the replication factor; otherwise, the default replication factor is set to three, which is standard for fault tolerance in existing distributed storage systems. The URL of the Big Data file source is also taken as input, as the enormous size of the Big Data file generally resides in cloud storage.

The proposed scheme gives the end user full flexibility to fragment the Big Data file according to any technique, such as vertical fragmentation, horizontal

fragmentation, or both. The main idea of horizontal fragmentation technique is to read records one by one (based on the chosen technique), buffer them in temporary buffers created on each distinct occurrence of records based on the selected attributes. The buffering of data takes place on the Master node and is sent to Datanodes when the buffer size reaches a particular threshold. The fragmentation algorithm is shown in the Algorithm 5.

Algorithm 5 Algorithm for Fragmentation

Require: $KeyAttributes[], ClusterIds[], bigFile, bufferSize, zone[[key1 : ClusteIds[]], [key2 : ClousterIds[]], ..., [keyn : ClusterIds[]]]$;
$ReplicationFactor = 3$;
while $recordinbigFile$ **do**
  if $record == isDistinctOnKeyAttributes[key]$ **then**
    create buffer[] of bufferSize in Masternode;
  end if
  while $size(buffer[])lessorequaltobufferSize$ **do**
    $insertrecordinbuffer[]$;
  end while
  while $bufferIsReady$ **do**
    if $clusterToSendnotFULL$ and $ClusterIdsisinzone[keyindex]$ **then**
      Dispatch bufferredData from buffer[];
      Update masterMetaData;
      Generate unique identifier for bufferredData;
      Store bufferredData in cluster;
      Update clientMetaData;
    else
      Find nearest clusterToSend from ClusterIds;
      Dispatch bufferredData from buffer[];
      Update masterMetaData;
      Generate unique identifier for bufferredData;
      Store bufferredData in cluster;
      Update clientMetaData;
    end if
  end while
end while

6.4.2 Identification generation phase

Buffered data is sent to the designated Datanodes whenever the buffer size reaches a specified threshold. These data are written into files (known as fragments) which

are then sent to the Datanodes. For each dispatch of buffered data to the Datanode, a unique identifier for that fragmented file is recorded in a JSON file on the Master node, known as master metadata. The fragment file names are created in such a way that they are unique among all the fragments and provide other relevant information about the Big Data file intelligently. The metadata on the Master node not only guides the deployment of computation to the desired Datanodes but also plays a vital role in failure recovery.

6.4.3 Allocation phase

The main challenge lies in pairing up the fragments with the Datanodes in such a way that even if a node fails, data can still be recovered. Generally, replication helps to achieve recovery smoothly, but in the proposed scheme, replication is used in a novel way that reduces the probability of data loss. As multiple key inputs are used to fragment the Big Data file, it is fragmented based on each key, and every set of single key-based fragments acts as a replica or image of other sets of different key-based fragments. In the proposed scheme, one particular key-based fragment of the Big Data file may not be guaranteed to be the exact image of other key-based fragments. The idea is not to keep the fragments as exact copies or images of others but to rearrange every copy based on the chosen key attributes and place them optimally so that no data is lost during a node failure.

So Let us assume that the set S is the set of all the records existing in the Big Data file. The Big Data file is fragmented and written into multiple files $k_1f_1, k_1f_2, k_1f_3, ..., k_1f_p$ based on one key attribute say k_1. Considering every fragment of $(k_1f_1, k_1f_2, k_1f_3, ..., k_1f_p)$ contains subsets of S, it is obvious that

$$k_1f_1 \cup k_1f_2 \cup k_1f_3 \cup .... \cup k_1f_p = S \tag{6.1}$$

Similarly consider the same Big Data file is fragmented based on the another key k_2 forming multiple files which are further assumed to be subsets as

$k_2f_1, k_2f_2, k_2f_3, ..., k_2f_q$.

It continues till the nth key forming subsets as

$k_nf_1, k_nf_2, k_nf_3, ..., k_nf_r$.

Let us assume that C_1 is the set of all Datanodes containing all the fragments which are fragmented based on the key k_1. Similarly C_2 is the set of all Client Nodes containing all fragments which are fragmented based on the key k_2 and this continues till C_n is the constituted with the set of Datanodes containing all the fragments based on key k_n. The main idea of the proposed scheme is to distribute all the key based fragments across the Datanodes such as

$$C_1 \cap C_2 \cap .... \cap C_n = \phi \tag{6.2}$$

The collection of Client Nodes containing all the fragments based on one key attribute is termed as *Zone*. According to Equation 6.2, no *Zone* stores fragments based on two different keys. Every *Zone* must contain all the fragments based on the same key so that even if a node of a particular *Zone* fails, all the fragments of any of the remaining *zones* are accumulated to form the Big Data file and that particular key based fragments of the failed node are regenerated by the node failure and detection algorithm itself ensuring failure recovery without data loss. The number of Zones is by default 3, but it grows if number of the chosen attributes are greater than 3. Every *Zone* can form exact image of other *zones*. Figure 6.3 depicts the allocation of fragments.

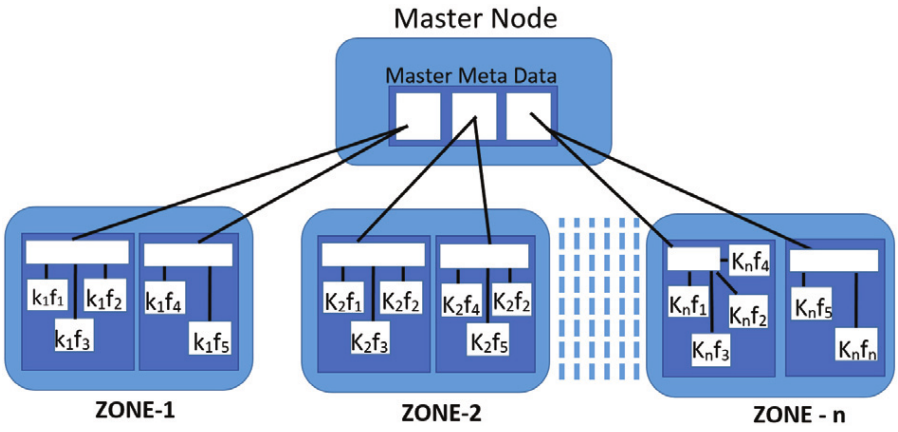

Figure 6.3: Fragmentation and allocation

6.4.4 Multilevel indexing phase

To improve the efficiency of the proposed system, multilevel indexing scheme is maintained. The Master meta- data is stored on Master node and the Secondary metadata are stored on those Datanodes where the fragmented data should be stored. The metadata structure is described in Section 6.3.3.

When the system starts reading the file from the cloud server, the system updates the Master metadata if there already exists the metadata file with respective file references, else new metadata file is created. While reading the file, the system creates buffers or data pools depending on the key(s) provided and continuously pushes the data into the buffer. Upon filling up the buffer, the system creates a data block from that buffer and sends to respective nodes and empty the buffer to continue the process. The Master metadata is updated with the secondary metadata reference.

The Datanode stores the data block in the respective directories. At the same time, it also updates the secondary meta file information with the datablock ID, clusterID, key details and fragmentation ID. If the metafile does not exist, system creates new secondary meta file with the help of cluster reference details. This process is repeated for other Datanodes depending upon the user-defined replication factor. The replication phase is described in the Section 6.4.3. Figure 6.4 presents the Flowchart of creating metadata for indexing.

6.4.5 Processing phase

In our proposed framework, the stored data are processed in parallel using MapReduce paradigm. In this system, like Hadoop, instead of running the code at all Datanodes in a cluster, the MapReduce [40] analytics code will run on some of the Datanodes. The system first checks which blocks of which nodes are necessary for the execution, then the execution process follows the split-apply-combine strategy, same as typical MapReduce programming paradigm on the Big Data set with parallel and distributed algorithms on a set of a cluster. The user provides the code written in Java or Python language through user interface. The code has some built-in functionalities to read the Datanode metadata file and start execution on

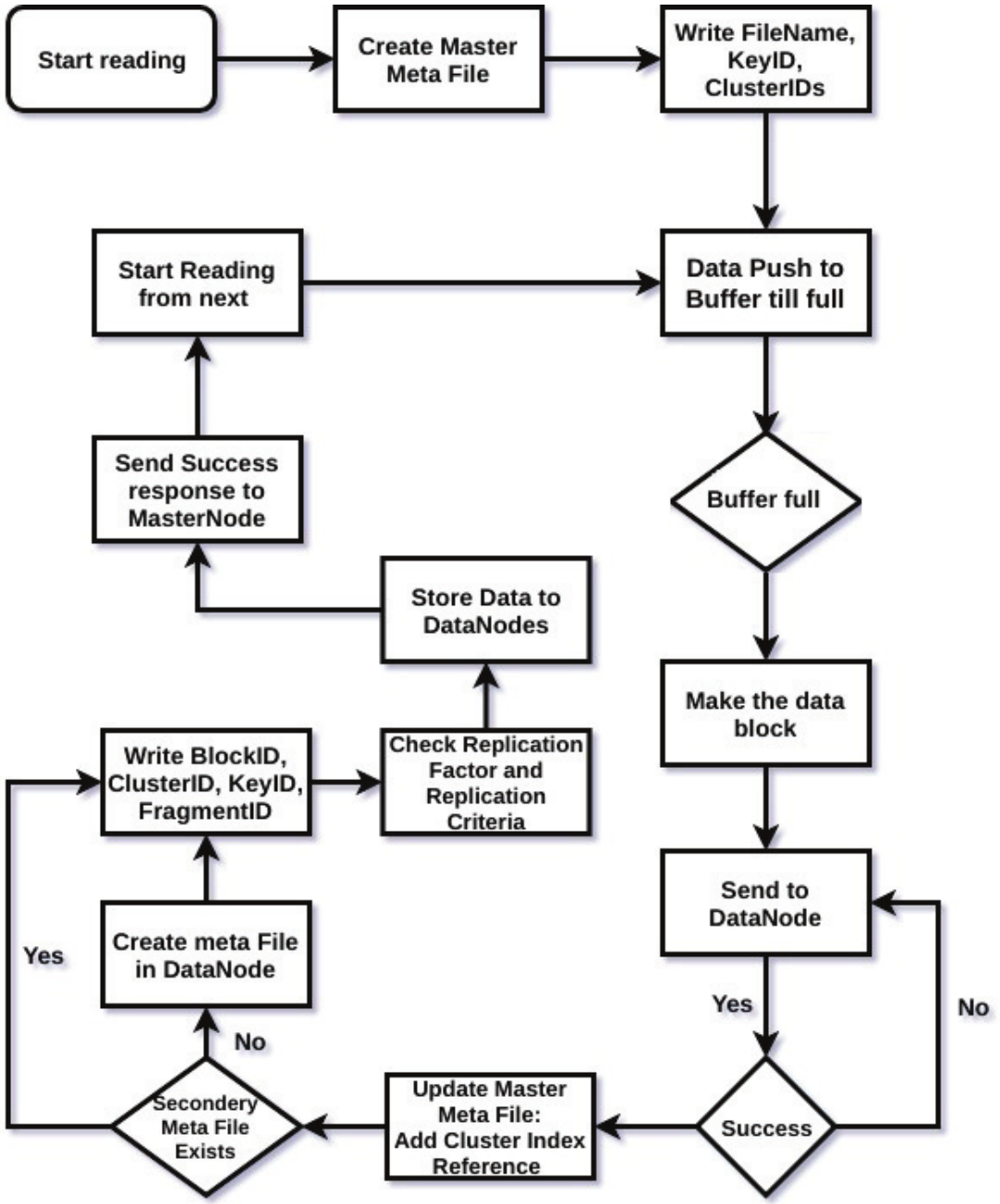

Figure 6.4: Fragmentation Allocation

the blocks, referred by the metadata file stored in the Datanode. The code picks up each block of data and starts parallel processing on the set of Datanode. The system continuously checks for the completion of the execution of the processes and accumulates (1st and 2nd phase of accumulation) the result. The user can view the result from the user interface.

The analytical phase of the proposed system is divided into six subphases:

- **Input format detection:** Input file detection is the first phase of analytics. The system detects the file type and sends the information to the executable code for break down the individual elements into touples.

- **Executable block detection:** This is the 2nd phase of the analytics process. During this process, the system makes a list of executable blocks by reading the metadata files. This means that the user given analytical code will take the listed nodes and blocks as input.

- **Map phase execution:** The map task works in the same way as the conventional MapReduce programming model, but map task runs only on the selected set of blocks of selected Datanodes.

- **Combine phase execution:** It is an optional phase, i.e. this phase may not be required for certain tasks. In this phase, the same types of key-value sets are combined together to make some new sets of fragments.

- **Reduce phase one (Executes inside the Datanode):** The first reduce phase runs on each Datanode which is already completed the map tasks. Once the execution is over, the zero or more key-value pairs ofdata is sent to the Master node for further reduction. This first phase of the reduction task is executed only inside the previously selected Datanodes.

- **Reduce phase two (Executes in master node):** In this phase, the application gathers all the data generated from the selected Datanodes and executes the final reduce task.

6.5 Useful Features

Here are some useful features of the NGMR system.

6.5.1 Fault tolerance

As all the fragments based on different keys belong to different zones, failure of a node containing one fragment can easily be formed from another zone. Yet there exists a little data loss probability. The data can only be lost if at least one node from every zone fails at the exact same time and every failed node contains the same set of records. Let us assume that failure probability of a particular Datanode in any zone be p and there exists $Z_1, Z_2, ..., Z_n$ *zones* ,each containing $(k + d)$ Datanodes where k denotes the minimum number of clusters needed to accommodate all the fragments of the Big Data file based on one key and d is variable for an additional number of Datanodes added to the same zone. Now, if d is ignored then the number of Datanodes of every zone remains the same to avoid further complications. As explained earlier, data can only be a permanent loss if at least one Datanode from every zone containing the same set of records or fails at the same time. The probability of this scenario to occur is (at least 1 node failure from zone Z_1) and (at least 1 node failure from zone Z_2) and ..... (At least 1 node fail from zone Z_n), which further signifies (1 - no of fails) and (1 - no of fails) and (1 - no of fails). . . so on. It can be deduced from the binomial distribution that the maximum probability of data loss can be deduced from the above statement is

$$1 - (K_0)p^0(1 - P)^K X1 - (K_0)P^0(1 - P)^K X...n^{th}term \tag{6.3}$$

which further implies the data loss probability as below.

$$1 - (1 - P)^K X1 - (1 - P)^K X... \tag{6.4}$$

which finally can be simplified as

$$1 - (1 - P)^{K^n} \tag{6.5}$$

Where K is the key factor or the replication factor. It may be concluded that the increasing number of *zones* or replication factors or key factor decreases the data loss probability drastically. It also signifies that a single Datanode failure will not result in any data loss.

6.5.2 Spatial locality

Spatial locality implies the tendency to access the nearest storage location repetitively once being referred to a particular location. In the proposed system as the main goal is to fragment and distribute the data based on the same key and all the queries are triggered based on the key, it can be shown that the same key-based data will be accessed repetitively. In the Datanodes, every *zone* consists of the same set of related data concerning key maintaining the nearest location at the cluster level. This key-based related data residency at every *zone* thus exploits the "Spatial Locality" at the cluster level of view. This approach help to achieve guided deployment of computation further reducing the control and communication overhead of computational costs.

6.5.3 Local execution

In a distributed system, lowering the usage of network bandwidth results in efficiency. The proposed system conserves network bandwidth by taking advantage of the fact that the input data is stored on the local disk of Datanodes as several copies of each block. Like Hadoop MapReduce, in the proposed system, the user programs are copied to multiple Datanodes. The master takes the location information of the input files into account and attempts to schedule a map task on a machine that contains a replica of the corresponding input data. Upon failure, it attempts to run a map task near a replica of the input data. The output data is accumulated in the Datanode with a reduce task first and then accumulated in the Master node and provides the final result. As the large dataset is read locally, it consumes no network bandwidth.

6.5.4 Status information and live node detection

The Master node runs a NodeJS server using the Express module and generates a set of status pages for user interaction. The status pages show the status of the *Live* and *Dead Nodes* by exchanging periodic heartbeat messages, progress of the MapReduce computation report, completed task and in-progress task details, document size, block size, bytes of intermediate data, bytes of output, processing rates, output

file location, file and key wise cluster distribution. The pages contain links to the run-time generated standard error and standard output. Unlike Hadoop, in the proposed system, the Master node sends a heartbeat message to the Datanodes after a short interval. In response, the Datanodes send a live status to the Master node. If the responses from the set of Datanodes are received, the Master node sends a special heartbeat message and waits for some time. If no response is obtained from those sets of Datanodes, the Master node updates its table of live and Dead nodes and changes its indexing and metadata accordingly. The flowchart of Live and Dead Node Detection in Figure 6.5.

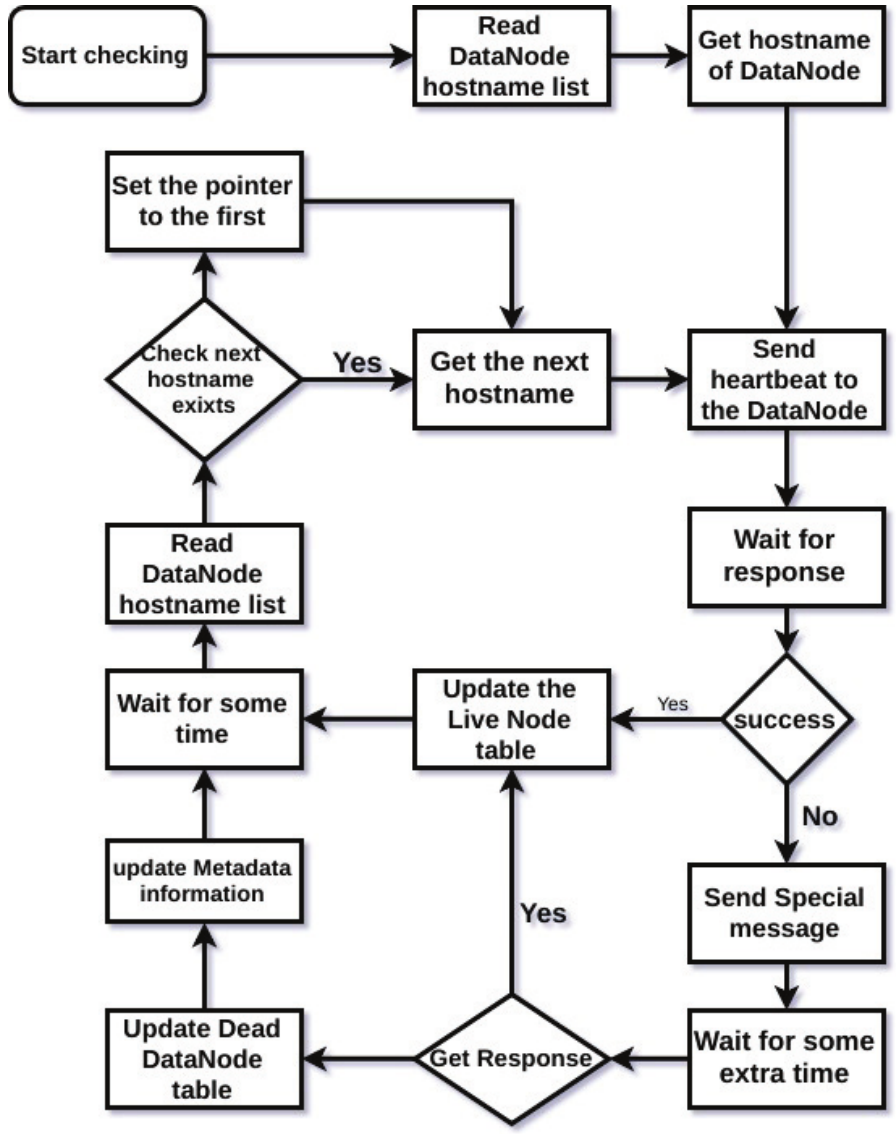

Figure 6.5: Detection of live and dead nodes

6.6 Setting up NGMR cluster

NGMR is developed with Node.js, a free, open-source server environment that runs on JavaScript on the server and can operate on various platforms, including Windows, Linux, and Mac OS. Based on the above discussion, a Raspberry Pi-based cluster has been deployed for this work. The advantage of this deployment is that, while experimenting with the NGMR file system, the effect of mobility on such systems is also being observed.

6.6.1 Data storage using Node.js

Handling large files is not new to JavaScript. In fact, in the core functionality of Node.js, there are several standard solutions for reading from and writing to files. The most straightforward solution is to stream the data in (and out). This option fails because the large set of data is not expected to be static data ("data at rest"). Hence event streaming is chosen for this work. The event streaming functions effectively for the situation where there is a constant flow of data ("data in motion") and the action needs to be taken in real-time as soon as possible.

6.7 Experimental setup

In this section, the experimental setup for the NGMR system is discussed. Before discussing the experiments, the data model used in this work and the data generation technique for carrying out the experiments are presented.

6.7.1 Data model

The data model used in this work is illustrated in Fig 3.3. However, data members are not shown here. A person can be a doctor or a patient. The patient can register one or more complaints regarding illness. A complaint is treated within one or more treatment episodes by doctors. A treatment episode may consist of one or more visits. During a visit, the doctor prescribes one or more investigations. Investigations

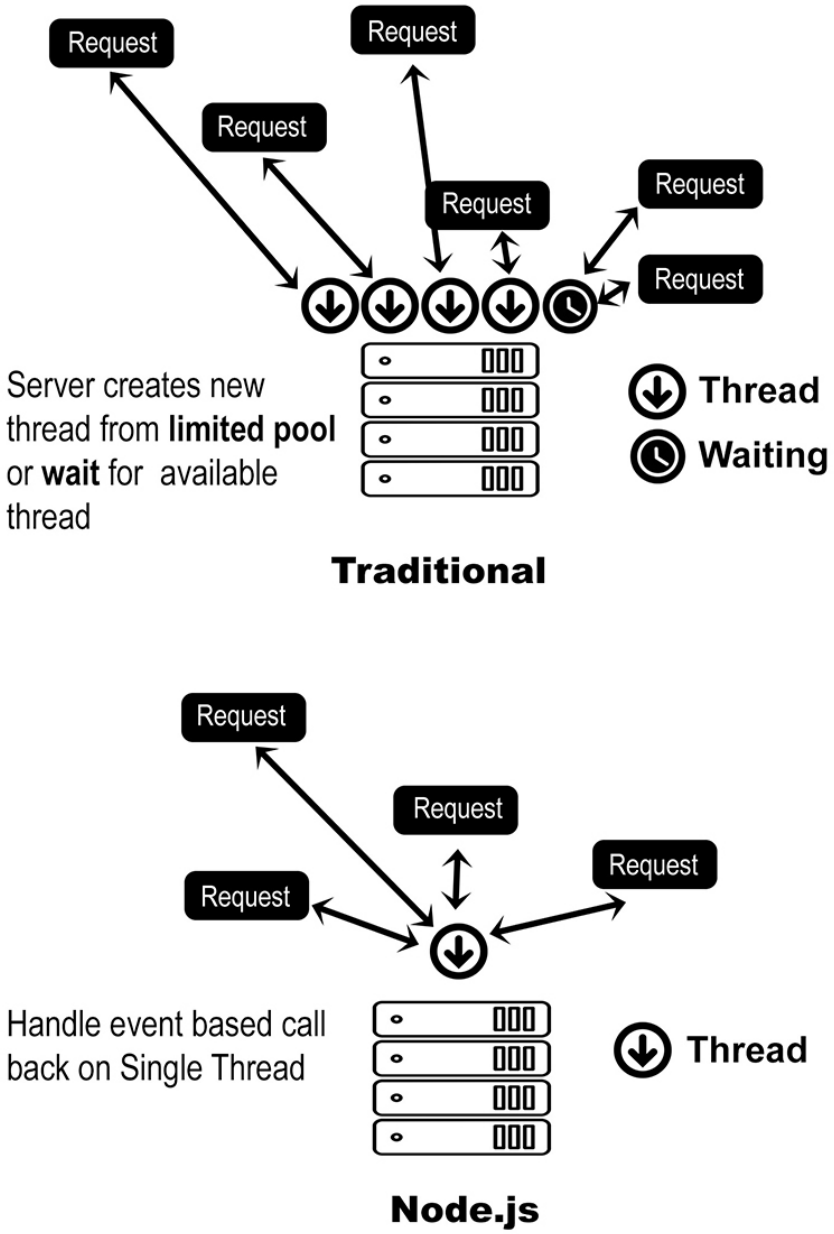

Figure 6.6: Node. js request handling thread structure

are of two types: continuous monitoring and discrete monitoring. Diagnosis can be concluded in a visit. More details of the data model are described in section 3.3.2.

6.7.2 Data generation

Kiosk-based health centers have been set up in rural areas, discussed in Appendix A. These centers are driven by medical professionals with the help of the cloud-based

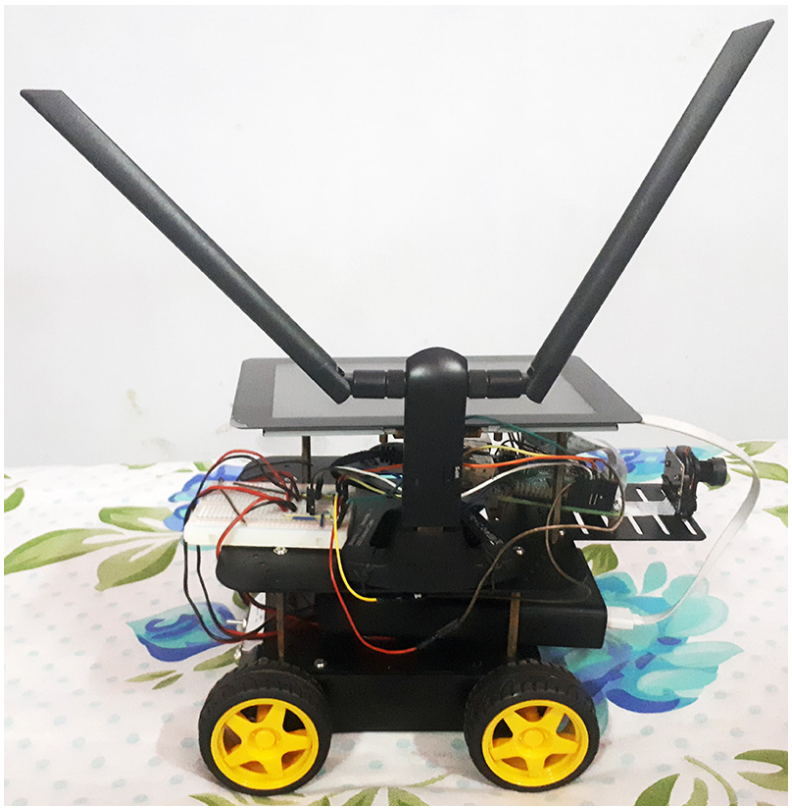

Figure 6.7: Replica of mobile node

application KiORH (Kiosk Operated Rural Health-care). The application provides an integrated environment for gathering symptoms and other information about patients visiting the kiosks and uploading the details to the cloud for real-time treatment by remotely located doctors. The data collected from rural patients through the application is stored in cloud storage. This collected data volume has been used for this research work. Although the data volume was not very large, it has been used to generate a large volume of synthetic data by duplicating, restructuring, and reusing. The synthetic data set is produced following the above-mentioned data model.

6.8 Execution of user-defined queries

An example of executing health-related queries using the aforementioned dataset may be considered. The user query is: 'Find the details of the patients who belong to Kolkata and are suffering from tuberculosis.' As the data model does not store

both pieces of information in the same file, two MapReduce jobs are run in Hadoop. First, Hadoop runs a MapReduce job to find the details of the patients who belong to the respective cities. These results are stored in the HDFS as output, and then the patient data is sorted and filtered by diseases. The overall execution time is the sum of the two MR jobs executed in Hadoop. However, the entire process requires a longer time due to the overhead mentioned earlier. Therefore, a better strategy is to discard the records containing key values other than those mentioned in the query either in the Map phase or in the Reduce phase.

6.8.1 Node.js query execution

In NGMR, the stored data is processed in parallel both across the Datanodes and within the Datanodes depending on their processor cores. In the NGMR framework, instead of running the MapReduce analytics code in all the Datanodes, the code runs in a small subset of the Datanodes. The NGMR system guides the MapReduce code to run in a specific set of selected nodes.

6.8.2 Query execution on Hadoop

The Hadoop framework performs parallel computation across a large cluster with a single Master node (JobTracker) and multiple Datanodes (TaskTrackers) using the MapReduce architecture. MapReduce has three basic components that perform distinct operations on the data: Mapper, Combiner, and Reducer.

When a user provides a query as a MapReduce (MR) Program, the JobTracker sends the user query (the MR Program) to all the Datanodes and provides the node block details to the framework. The JobTracker monitors the progress of the MapReduce task and coordinates the execution of map and reduce tasks on the TaskTrackers, which run on every node. It can happen that the user-given query will not return any data after the execution of the MapReduce job from a particular node; still, the TaskTracker will run and initiate the reduce task. The task returns a zero-reduced set of data. This process increases the total execution time of the MapReduce task. In Figure 6.8, the end-user needs details

for a key-value $k1$. In this case, the MapReduce engine will run on all blocks on all nodes, as shown in Figure 6.8.

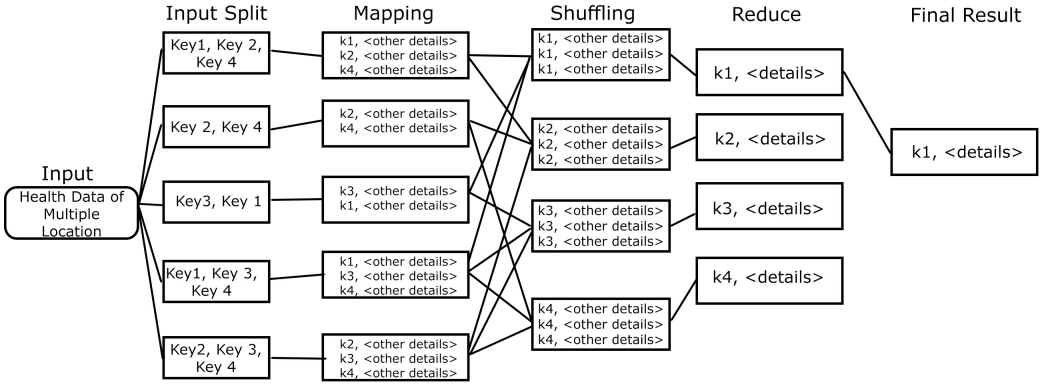

Figure 6.8: Hadoop MR Process on health records of multiple cities

6.8.3 Query execution with guided MapReduce

When a user places a query in the NGMR framework, the NGMR Master node checks which blocks on which nodes are really necessary for the execution of the given MR query. The framework then follows the split-apply-combine strategy in the same way as a typical MapReduce programming paradigm on the data blocks distributed in the cluster. The user provides the code written in Node.js. The framework reads the query and extracts the key attributes and their values required for executing the query. The Master node then reads the master metadata stored for the file referred to by the user. From the master metadata, the Master Node gets the references of the Datanodes along with the metadata stored in those Datanodes. The framework reads the Datanode metadata referred to by the Master node and gets the references of the data blocks that need to be processed for the execution of the user-given query. The framework then sends the MR code to the Datanodes where the requested data blocks are stored. The map task is executed in the same way as the conventional MapReduce programming model but works only on the selected Datanode blocks.

After the map process, in the NGMR framework, the reduce task is executed in two phases. The first phase is executed in each Datanode that has already completed

the map task, and the second phase takes place inside the Master node. Once the first phase of the reduce task is completed, the key-value pairs of output data are sent to the Master node for further reduction. After the second phase of the reducing tasks, the framework stores the output data as final results in the NGMR file system. Figure 6.9 shows the guided MapReduce process in the NGMR framework.

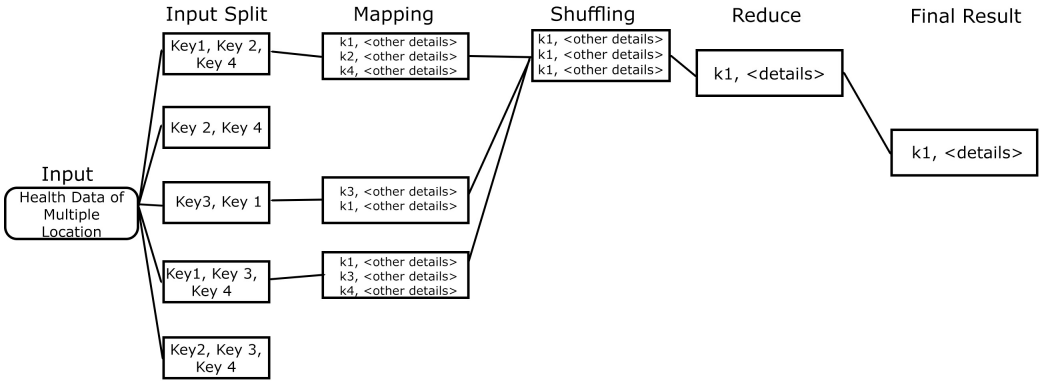

Figure 6.9: NGMR framework MR process on health records of multiple cities

6.8.4 Parallel execution in processor core

NGMR uses an NPM package called ‘mapred‘, which follows the Google MapReduce specification. The input should be a key-value paired array like ‘map(key1, value1)‘, so that a set of intermediate key-value lists ‘(key2, value2)‘ are created for reduction. The reduce program accepts the intermediate key and a set of values for that key, like ‘reduce(key2, list(value2))‘, and produces a smaller set of values like ‘list(value2)‘. The ‘mapred‘ package has the provision to use all cores of the processor, or the user can specify how many cores should be used by the system to process the MapReduce code.

The operating system runs the I/O in parallel and sends the data to single-threaded JavaScript. The NGMR code consists of small portions of synchronous blocks that run quickly and pass the data to files and streams. Therefore, the JavaScript code does not block the execution of other pieces of JavaScript. More time is spent waiting for I/O events to happen than executing JavaScript code. The NGMR Node.js framework just invokes functions and does not block the execution

of other code. It gets notified through callbacks when previous code execution is over, and the system receives the result. Node.js does not evaluate the next code block in the event queue until the previous one finishes executing. So, the code is split into smaller synchronous code blocks and call the callback function is called to inform Node.js that the previous code execution is over, allowing it to continue executing the pending tasks in the queue.

6.9 Discussion

Consider a user query: 'Find the details of the patients who belong to Kolkata and are suffering from tuberculosis.' As the data model does not store both pieces of information in the same file, two MapReduce (MR) jobs need to be executed. First, Hadoop runs an MR job to find the details of the patients who belong to the respective cities. These results are stored in the HDFS as output, and then the patient data is sorted, and the output is filtered by diseases. The overall execution time is the sum of the two MR jobs executed in Hadoop. However, the entire process requires more time due to the overhead mentioned earlier in Section 6.1.

The NGMR system is developed using multiple nodes. One node is set up as the Master node, and the other four nodes are used as Datanodes. In the setup used for the research work four Datanode are used. The machines are arranged in a two-level tree structure switched network. All the machines are in the same network attached to a switch. Out of the total hard disk space, 20% is reserved for the OS and application programs, and 80% is for storage and processing.

Generally, the block size is 512 bytes. But for Big Data, each chunk size may grow up to 128 MB . One chunk will be represented as one of the blocks in a distributed file system. For example, one chunk (aka one block of Big Data, 128 MB) is known as a big block. To accommodate one big block,

$$\frac{128 \text{ MB}}{512 \text{ B}} = 256,000 \text{ blocks}$$

is needed.

Therefore, every 1 TB hard drive of the Client node can accommodate 8,000 big blocks. To fragment a Big Data file of size 512 TB into 128 MB chunks. The number of big blocks needed is:

$$\frac{512 \text{ TB}}{128 \text{ MB}} = 4,000,000 \text{ blocks}$$

To accommodate 4 million big blocks, 1 trillion blocks need to be allocated on the underlying hard drive disk by the OS. Blind partitioning of Big Data can result in a significant cost in terms of block access. The fewer blocks accessed, the less time consumed, and read-write delay decreases drastically. Blind partitioning can force the OS to access nearly one trillion blocks for a 512 TB file.

In the Hadoop Distributed File System (HDFS), Hadoop does not use metadata for tracking the data blocks of a file. It uses another way of indexing, storing the reference of the next part of the data block at the end of the previous block.

Suppose a 512 TB Big Data file contains health-care data from seven cities in India. For Kolkata and Hyderabad, the data size is 256 TB (128 TB each), for Pune, Bangalore, and Chennai, the data size is 192 TB (64 TB each), and for Mumbai and Delhi, the data size is 64 TB (32 TB each). The percentage of big blocks needed per city is:

- Kolkata: 25%
- Hyderabad: 25%
- Pune: 12.5%
- Bangalore: 12.5%
- Chennai: 12.5%
- Mumbai: 6.25%
- Delhi: 6.25%

By maintaining a city-wise index table, a maximum of 25% and a minimum of 6.25% of the one trillion blocks per city need to be accessed. Blind partitioning may lead to accessing one trillion blocks for each city every time in the worst case.

Spatial locality helps to achieve fewer block accesses and faster information retrieval. The unique identification generation scheme results in efficient mapping of the fragmented blocks in the respective Datanodes. This aids in guided deployment of analytical MapReduce queries on specific Datanodes, rather than deploying them on all nodes.

6.10 Summary

In this chapter, the storage and retrieval of data in a distributed system in Big Data environment is discussed. A novel framework called *Node Guided Map Reduce* is proposed to overcome the challenges faced in Hadoop and other distributed systems and to reduce the query response time. An application for the framework is developed. The application decreases the retrieval and execution time of Big Data files in a distributed pattern by not involving all the nodes in a cluster. The information of the nodes where the data should be stored is collected from the user. The data is fragmented using horizontal and vertical fragmentation algorithms. After deploying the application on a Raspberry Pi, it was also deployed on a high-end cluster. In the next chapter, the comparative analysis of various Big Data tools along with the NGMR application is presented.

Without big data analytics, companies are blind and deaf, wandering out onto the web like deer on a freeway.

— Geoffrey Moore, American consultant and author

7 Comparative analysis of Big Data tools

Contents

7.1 Prelude

This chapter aims to present some observations regarding the selection of NoSQL systems along with the Big Data tools which are appropriate for a health-care application. For this purpose, a standard data model is used and a set of nine queries are selected for the execution of different database operations. This allows a user to understand the performance of databases better and to find out which system is better for a particular workload.

This chapter explores the challenges of storing and managing healthcare data using Big Data tools and NoSQL databases, highlighting the limitations of traditional relational databases in handling the vast, diverse, and fast-moving nature of health data, which includes structured, semi-structured, and unstructured formats. Given the critical need for timely and accurate query responses in services like remote consultation, patient monitoring, and transaction processing, the study evaluates the performance of platforms such as Hadoop, HIVE, Cassandra, MongoDB, HBase, and NGMR. A comparative analysis was conducted to assess their ability to efficiently store and retrieve health-related data. The chapter underscores the importance of selecting appropriate Big Data solutions based on application-specific requirements, as query execution time and storage efficiency vary across platforms. It concludes that a thorough understanding of the underlying data models and query structures is essential for optimizing health data storage and retrieval, providing valuable insights into the suitability of platforms like above said tools for health-care applications.

The major requirement for the experiment is a data model for storing data and a set of queries which is required to be executed on a multi-node cluster to observe the performance of each NoSQL system and tool. The following sections present the data model and details of the experimental setup and the methodology adopted for the experiment. The results of the experimets are presented in Section 7.4.

The experiment is designed following YCSB guidelines, offering valuable insights into the applicability of Big Data solutions in the health-care domain.

7.2 Data model

Initially, a data model was built for primary health-care delivery 3.3. Data modelling for the healthcare domain is difficult because the healthcare domain is both broad and deep. The number of data concepts and ways in which they can relate to one another is shown in the Figure 2.1. There are two different participants, such as patients and physicians. Each participant has a different view of the care processes they are involved in.

After incorporating the participants, problems arise due to the convoluted nature of clinical knowledge and how very few processes are perfectly repeatable across time and patient populations. As the main objective is to model primary health-care, the same data model is used as described in Section 3.3.2. Figure 3.3 shows the used data-model.

In this section the data model structure is described as a Class Diagram, the attributes are listed below to visualize the design of the object-oriented structure of the data model.

Attributes	Description
pt_id	Stores a unique Patient ID
pt_name	Stores Patient Name
pt_dob	Stores Patient Date of Birth
alternate_pt_ids	Stores Patient's alternative IDs if he/she is admitted to other organizations with different patient IDs
pt_gender	Stores Patient Gender
pt_occupation	Stores Patient Occupation
pt_address_type	Stores Patient Address Type (whether it's permanent or not)
pt_address	Stores Patient Address
pt_city_town_vill_ps	Stores Patient City/Town/Village Name
pt_district	Stores Patient's District
pt_state	Stores Patient's State
pt_pin	Stores Patient's Pin Code
pt_country_code	Stores Patient's Country
pt_email_id	Stores Patient's Email ID
pt_phone	Stores Patient's Phone Number
pt_phone_2	Stores Patient's 2nd Phone Number (if any)
pt_mob	Stores Patient's Mobile Number
ecp_name	Stores Contact Person of the Patient in Case of Emergency
ecp_relation	Stores Relation between the Patient and Emergency Contact Person
ecp_address_type	Stores Address Type of the Emergency Contact Person (whether it's permanent or not)
ecp_address	Stores Address of the Contact Person
ecp_city_town_vill_ps	Stores City/Town/Village Name of Emergency Contact Person
ecp_district	Stores District of Emergency Contact Person
ecp_state	Stores State of Emergency Contact Person
ecp_pin	Stores Pin Code of Emergency Contact Person
ecp_country_code	Stores Country of Emergency Contact Person
ecp_email_id	Stores Email Address of Emergency Contact Person
ecp_mob	Stores Mobile Number of Emergency Contact Person
history	Stores Patient's Present, Past, Personal, Family, and Immunization History
socio_economic_status	Stores Patient's Socio-economic Status
pt_food_habits	Stores Patient's Food Habits (e.g., veg, non-veg, or others)
pt_religion	Stores Patient's Religion

Table 7.1: Attributes of Patient Class

Attributes	Description
pt_id	Store a unique Patient ID
doctor_id	Store a unique ID for every Doctor
doctor_name	Stores Doctor name
dr_type	Primary physician / Consultant / Specialist / Dental Surgeon / Orthodontist / (Nurse / Physiotherapist for care_provider)
dr_specialization	Store Doctor Specialization
dr_organization	Store organization name or code, where the respective Doctor is attached to
dr_address_type	Stores the address type of the Doctor, i.e., permanent address or not
dr_address	Stores Doctor Address
dr_city_town_vill_ps	Stores Patient City/ Town/ Village
dr_district	Stores Doctor district
dr_state	Stores Doctor State
dr_pin	Stores Doctor pin code of address
dr_country_code	Stores Doctor Country
dr_email_id	Stores Doctor Email ID
dr_mob	Stores Doctor Mobile
dr_phone	Store doctor phone number
*dr_phone*2	Store doctor another phone number

Table 7.2: Attributes of Doctor Class

Attributes	Description
treatment_id	Stores a unique ID for each treatment
Pt_id	Stores patient ID
Pt_name	Stores Patient Name
Pt_addr	Stores Patient Address
Pt_gender	Stores Gender
Pt_fd_hbt	Stores patient food habit
Pt_allergy	Stores patient allergy details
Blood_grp	Stores patient Blood group
Dr_id	Stores doctor ID who is responsible for current treatment
dr_name	Stores doctor name who is responsible for the current treatment of respective patient
dr_specialization	Stores doctor specialization
start_date	Treatment start date
is_active	Is this treatment still active or not
complaint	Stores patient complaint description
diagnosis	Stores diagnosis description by Doctor
Clinical_status	Stores current clinical status of the treatment

Table 7.3: Attributes of Treatment Class

Attributes	Description
visit_id	Stores unique visit ID
pt_id	Stores patient ID as a reference
pt_name	Stores Patient Name
pt_add	Stores patient address
pt_food_habbit	Stores patient food habit
allergy	Stores Patient allergy details
blood_grp	Stores patient blood group
treatment_id	Stores unique treatment ID
trt_start_date	Stores treatment start date
initial_complaint	Stores basic complaint
visit_time	Stores visit time stamp
reason	Stores reason for the visit; e.g., any new complaint or check-up, etc.
systolic_bp	Stores systolic BP
diastolic_bp	Stores diastolic BP
pulse_rate	Stores pulse rate
temp	Stores body temperature
respiration_Rate	Stores patient's respiration rate
height_cms	Stores patient height
weight_kgs	Stores patient weight for every visit
investigations	Stores the tests proposed by the doctor
invtn_reports	Stores the results of the tests
medication	Stores the medicines prescribed by the doctor and their results as key-value pairs
Diagnosis	Stores the disease diagnosed by the doctor
therapies_surgeries	Contains several therapies and surgeries undergone along with the date
current_status	Stores the current status

Table 7.4: Attributes of Visit Class

Attributes	Description
pt_id	Stores Patient ID
trt_id	Stores Treatment ID
visit_id	Stores Visit ID
parameter	Stores the parameter which the sensor measures
data	Stores all the measured data along with timestamp

Table 7.5: Attributes of Sensor Data Class

7.3 Experimental setup

For the experiment setup, the following things are followed to understand the performance of the tools:

7.3.1 File structure for data storage

The Health record data model discussed in the previous section is converted to a structural form for analysis of data through Hadoop. The attributes of the classes are therefore converted to XML as these files have a regular orientation. Every class is stored as a separate file. Every 'Row-key' which is the attribute of the class, is made an XML-element, also called XML-node or XML-tag of XML file. Following is the example of how the patient class is converted to XML 7.1 for analysis in Hadoop using MapReduce.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
<pt_id> Patient Unique ID </pt_id>
<pt_name>Name </pt_name>
<pt_dob> Date of birth </pt_dob>
<alternate_pt_ids>Alternative ID separated by comma
</alternate_pt_ids>
<pt_gender>  Gender </pt_gender>
<pt_occupation> occupation </pt_occupation>
<pt_address_type>Address Type; Permanent or current</pt_address_type>
<pt_address> Address </pt_address>
<pt_city_town_vill_ps>City, town, village Name </pt_city_town_vill_ps>
<pt_district>  district </pt_district>
<pt_state>  state </pt_state>
<pt_pin>  Pincode </pt_pin>
<pt_country_code> country </pt_country_code>
<pt_email_id>Email ID, if exist, else left blank</pt_email_id>
<pt_phone_l>  phone number </pt_phone_ll>
<pt_mob>  Mobile Number</pt_mob>
<Ecp_name>emergency contact person name</Ecp_name>
<Ecp_relation>relation with that contacted person</Ecp_relation>
<Ecp_address_type>Emergency contact person address type</Ecp_address_type>
<Ecp_address>Emergency contact person address</Ecp_address>
<ecp_city_town_vill_ps>
Emergency contact person city/ town/ village name
</ecp_city_town_vill_ps>
<ecp _district>Emergency contact person district</ecp _district>
<ecp _state>Emergency contact person state</ecp _state>
<ecp _pin>Emergency contact person pin code</ecp _pin>
<ecp _country_code>Emergency contact person country</ecp _country_code>
<ecp _email_id>Emergency contact person email address</ecp _email_id>
<ecp_mob>Emergency contact person mobile no</ecp_mob>
<history>
Present, Past, Personal, Family, Immunization History
</history>
<socio_economic_status>Socio economic status</socio_economic_status>
<pt_foodhabbit>Food habit</pt_foodhabbit>
<pt_religion>Religion</pt_religion>
<properties>
```

Listing 7.1: The details of patients

This block is repeated multiple times. One 'properties' tag holds single patient information, separated by its 'Row-key' name tag (Listing 7.1). In the same way, the Doctor class converted to XML file for analysis of the doctor data as shown in Listing 7.2. So large volumes of doctor data are parsed in the same way.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
<dr_id> Stores a unique doctor ID </dr_id>
<dr_name> Stores doctor name </dr_name>
<dr_type >
Primary physician / Consultant / Specialist / Dental Surgeon /
Orthodontist / (Nurse/physiotherapist for care_provider
</dr_type>
<dr_specialization>
Store Doctor Specialization
</dr_specialization>
<dr_organization>Store organization name or code, where the
 respective Doctor is attached to
</dr_organization>
<dr_address_type>
Stores the address type of the Doctor, i.e. permanent address or not
</dr_address_type>
<dr_address> Stores Doctor Address </dr_address>
<dr_city_town_vill_ps>
Stores Patient City/ Town/ Village Name of Doctor
</dr_city_town_vill_ps>
<dr_district> Stores Doctor district </dr_district>
<dr_state> Stores Doctor State </dr_state>
<dr_pin> Stores Doctor pin code of address </dr_pin>
<dr_country_code>Stores Doctor Country</dr_country_code>
<dr_email_id> Stores Doctor Email ID </dr_email_id>
<dr_mob> Stores Doctor Mobile </dr_mob>
</properties>
```

Listing 7.2: The details of doctor

This block is repeated multiple times similar to the patient file. One *properties* tag holds single doctor information, separated by its *Row-Key* name tag. The treatment (Listing 7.3), visit (Listing 7.4), sensor data(Listimng 7.5) classes are converted to XML files which are shown bellow.

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
<treatment _id>
Stores A unique ID for each treatment
</treatment>
<pt_id> Stores patient ID </pt_id>
<pt_name> Stores Patient Name</pt_name>
<pt_addr> Stores Patient Address</pt_addr>
<pt_gender> Stores Gender </pt_gender>
<pt_fd_hbt> Stores patient food habit </pt_fd_hbt>
<pt_allergy> Stores patient allergy details </pt_allergy>
<blood_grp> Stores patient Blood group </blood_grp>
<dr_id>
Stores doctor ID who is responsible for current treatment
</dr_id>
<dr_name>
Store doctor name who is responsible for the current treatment of the respe
</dr_name>
<dr_specialization>
Stores doctor specialization
</dr_specialization>
<start_date> Treatment start date </start_date>
<is_active>
Is this treatment still active or not
</is_active>
<complaint>
Store patient complaint description
</complaint>
<diagnosis>
Stores diagnosis description by Doctor
</diagnosis>
<clinical_status>
Store the current clinical status of the treatment
</clinical_status>
</properties>
```

Listing 7.3: The details of treatment

```
      <properties>
<visit_id> Stores unique visit ID </visit_id>
<pt_id>  Stores patient ID as a reference </pt_id>
<pt_name> Stores Patient Name </pt_name>
<pt_add> Stores patient address </pt_add>
<pt_food_habbit>
Stores patient food habit
</pt_food_habbit>
<allergy> Store Patient allergy details </allergy>
<blood_grp> Store patient blood group </blood_grp>
<treatment_id> Stores unique treatment ID </treatment_id>
<trt_start_date>
Store treatment start date
</trt_start_date>
<initial_complaint>
Store basic complaint
</initial_complaint>
<visit_time> Stores visit time stamp </visit_time>
<reason >
Stores Reason for the visit; e.g.- any new complaint or check-u etc.
</reason>
<systolic_bp> Stores systolic Bp </systolic_bp>
<diastolic_bp> Stores diastolic BP  </diastolic_bp>
<pulse_rate> Stores pulse rate </pulse_rate>
<temp>   Stores body temperature </temp>
<respiration_Rate>
Stores  respiration rate of the patient
</respiration_Rate>
<height_cms> Stores patient height </height_cms>
<weight_kgs>
Stores patient weight at every visit
</weight_kgs>
<investigations>
Stores the tests proposed by the doctor
</investigations>
<Invtn_reports>
Stores the results of the tests
</invtn_reports>
<medication >
Stores the medicines prescribed by the doctor and their
results as key-value pair;
</medication>
<Diagnosis>
Stores the disease diagnosed by the doctor
</Diagnosis>
<therapies_surgeries>
Stores Contains several therapies and surgeries
undergone along with the date
```

```
</therapies_surgeries>
<current_status> Current status </current_status>
</properties>
```

Listing 7.4: The details of visit

```
      <properties>
<pt_id> Stores Patient ID </pt_id>
<trt_id> Stores treatment ID </trt_id>
<visit_id> Stores visit ID</visit_id>
<parameter>
Stores the parameter that the sensor measures
</parameter>
<data>
Stores all the measured data along with the timestamp
</data>
</properties>
```

Listing 7.5: The details of sensor data

7.3.2 Set of queries

With the help of a large volume of data recvords generated from the data model, a comparison is made between the data processing time and the amount of data stored in the various data storages. The remaining part of this chapter studies the processing time of a large volume of data against a varied number of nodes when data volume is fixed. Another set of experiments are carried out by changing the data volume while keeping the note count fixed.

There are nine queries which are processed after customization for each of the tools. The queries are given below.

- Find all doctor's names and their details with specialization string 'cardiologist';
- Find all information of patients like demographic info, religion, food habits etc, with patient's name 'XYZ';
- Find all ECG reports of a particular patient with name 'x';
- Find history of 'Diabetes' of a particular patient with name 'x';
- Find sensor observed data of a patient for one hour;

- Patient treatment case history with name 'B';
- Find all the doctor's names who are attached with the organization whose id is "ABC-XYZ";
- Find all the complaint details of Patient with name 'B';
- Number of babies, whose age between 0 to 5, suffering from malnutrition;

Let us take an example of the second query: *Find all information of patients with patient's name 'xyz'.* For this query,

- it is required to write a MapReduce Java or Python code in the case of Hadoop.
- In the case of Hive, the query is written in Hive Query Language (HiveQL). The query is like "select * from the patient where name='xyz';".
- In the case of HBase, HBase command are need to write in HBase Shell. The command is like "get 'patient',{COLUMN $\Longrightarrow$ 'name:xyz'} ".
- For Cassandra, Cassandra Query Language (CQL) is used in the Cassandra query language shell (cqlsh). The syntax of CQL is the same as HiveQL. To execute the first query, the CQL will be "SELECT * FROM patient WHERE name=xyz;".
- In the case of MongoDB, MongoDB system-defined methods are used. The query is: "db.patient.findOne(name: "xyz")".
- The query (Cypher) for Neo4j to find all the complaint details of the patient with the name 'xyz': Match (p: Person name: "XYZ") RETURN p;

7.3.3 Cluster setup

The servers used in the experiment are deployed in a laboratory setup. A test bed is prepared with seventeen desktop computers, i.e. sixteen low-end and one high-end desktop computer has been prepared . The 16-node cluster configuration is as follows:

The master node is a high-end system HP Proliant ML350 G6 Server with 32 GB RAM, and 600 GB SAS HDD with 16 core processors. The client nodes are computers, each with quad-core processor with 8GB RAM and 1TB hard disk. The setup is shown in Figure 7.1.

Figure 7.1: 16 node cluster

7.3.4 Methodology

For this kind of experiment, there is a need for unstructured, semi-structured health data. So, multiple no of kiosks [102] in the rural villages ware setup. The

ICT-based cloud storage continuously stores the incoming health data from the kiosks. Those data are not enough for the experiment purposes, So, based on that data, lots of synthetic data has been generated.

There are sixteen slave nodes Figure 7.1 connected with a 32 port switch. Each node is installed with Ubuntu 20.04 as OS and the master node has CentOS7. The versions of the tools are as follows:

Hadoop: version 3.3.0

Hive: version 3.1.3

HBase: version 2.3.4

Cassandra: version 4.0

MongoDB: version 6.0

NGMR: vesrion1.1

The health data is stored in HDFS for the experiment using Hadoop, Hive and HBase. The MapReduce code was written for each query. For Cassandra, MongoDB, and NGMR, the code is written code using Python, PHP and NodeJS respectively for fetching the data from the cloud to the tool storage.

Before starting the experiment using each tool, the old data has been deleted to overcome the overlapping issue. Each query is executed a minimum of 3 times, and the average of those results is taken for plotting the graph. The experiment is started with 10 patient records of 149.8 kb size and continued up to 3 lakh records of size 197 GB. The default configuration has been used for each tool while experimenting.

Each time a query is executed starting from 1 node to 16 nodes. The node count is increased to 1, 2 4, 8, and 16. The experimental results are discussed in Section **??**.

7.4 Results and discussion

Based on the above experimental setup, a study is made on the data retrieval time while executing the query. In this section the results of a three-set experiment are discussed: (i) varying the number of cluster nodes for each tool and observing the effect; (ii) varying the amount of data while keeping the number of node count fixed

for each tool and observing the effect; and (iii) observing the query execution time by executing each query on 6 different tools and comparing the execution times. All nine queries are run. Due to the space limitation, only query 9 is presented in this chapter.

The first and second sets of experiments are conducted using all six tools. The results are plotted on separate charts for better understanding.

7.4.1 Performance of the Big Data solutions

In this section, performances of the Big Data solutions are shown on multi-node clusters.

Performance of Hadoop It is observed that while executing the queries in Hadoop, the use of a higher number of slaves does not imply better performance. Optimal performance is obtained when a particular load is given to a particular number of slaves. It is also observed that performance with 16 slaves improves and becomes better in comparison with others when the number of records is higher as shown in Figure 7.2. The experimental data is displayed in Table 7.6

Dataset Size	Node 1	Node 2	Node 4	Node 8	Node 16
1000	104153	90682	59370	47587	90818
3000	111310	97277	68667	52813	96753
4000	71964	88486	73677	89243	111600
5000	118468	103871	77965	58040	102687
8000	80926	150971	95683	98468	133887
10000	136361	120358	101208	71105	117524
12000	15568	132751	95841	101418	167460
15000	154255	136844	124452	84170	132360
16000	139459	149497	105608	100154	117411
20000	120381	150619	111280	112364	153512
25000	190042	169817	170939	110301	162032
30000	207936	186304	194183	123367	176869
40000	297069	205641	166829	113234	118649
50000	279510	252250	287257	175628	236214
60000	381067	240230	122535	121291	272765
70000	351085	318196	381131	227890	295559
80000	512431	208762	168189	153135	224489
90000	422659	384142	473106	280152	354904
100000	564309	412124	529942	148772	394488
110000	494233	450088	566080	332413	414249
120000	519326	483062	705074	286966	318924
125000	547914	499548	635811	371610	458757
140000	679526	549007	585904	616226	516212
150000	637382	581980	752029	436937	532939
160000	768653	690845	1261606	717787	747145
175000	726850	664414	868247	502264	607120
180000	800124	680899	1879838	500288	637933
200000	829877	906515	781437	409229	698794
250000	975623	911712	944381	607626	991249
300000	987651	978729	1112324	912223	799591

Table 7.6: Hadoop Execution Time (All time in ms)

It is noted that there is a sudden rise in the execution time when 4 Datanodes are used and the record count is 1.8 lakhs. This may be caused due to network latency. At this moment the executed result takes a longer time to arrive at the master node and for that reason, the overall execution time is increased.

Performance of Hive Similar results are observed for Hive. Performance on 16 slave nodes improves and becomes better in comparison with others when the number of records is higher as shown in Table 7.7 and Figure 7.3.

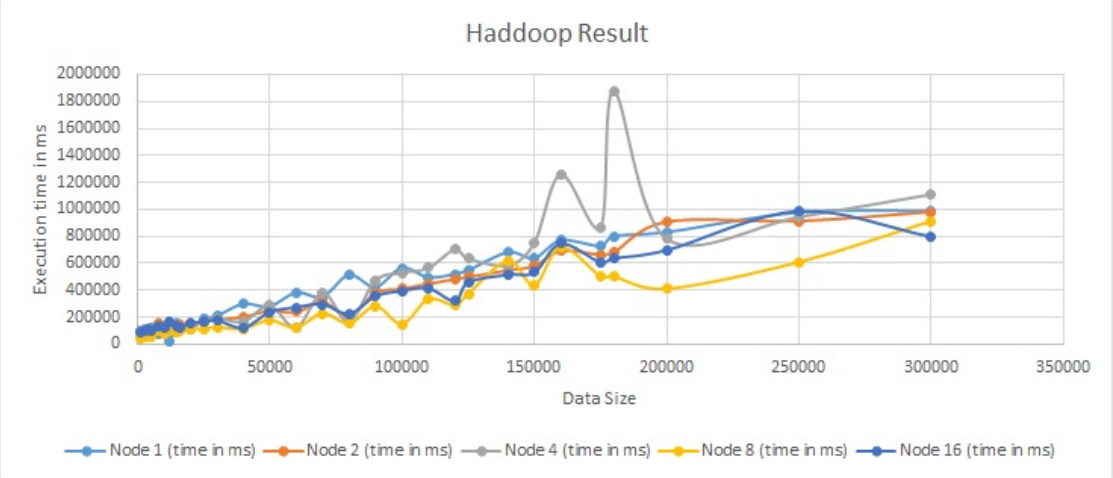

Figure 7.2: Hadoop execution time 1-16 nodes

Dataset Size	Node 1	Node 2	Node 4	Node 8	Node 16
1000	120	182	429	786	1241
3000	376	216	671	1992	1988
4000	421	498	923	2122	3721
5000	563	630	1022	3298	4681
8000	957	1160	2874	5622	6320
10000	1001	1867	3214	6673	8621
12000	1125	2845	5721	8632	9743
15000	1460	2969	7869	9821	9991
16000	1793	3251	9762	11852	12934
20000	2183	7532	14845	15605	18221
25000	772	5063	4691	14275	11231
30000	6401	10637	10632	20239	16963
40000	4750	13974	20237	19521	21341
50000	28918	32934	34395	44096	39889
60000	12340	18982	26710	32611	27558
70000	51435	55230	58159	67952	62815
80000	29546	34763	38246	51426	48423
90000	73951	77526	81922	91809	85742
100000	50807	53870	54718	71991	76061
110000	96468	99822	105686	115666	108668
120000	99332	109681	96134	120467	98833
125000	113356	116544	123508	133558	125862
140000	120965	131550	122478	154759	117582
150000	141502	144414	153212	163379	154520
160000	153710	150937	153261	195744	157444
175000	169648	172284	182917	193200	183178
180000	189782	189717	187606	218321	213246
200000	224094	229023	228137	244735	245940
250000	268292	270352	309319	292346	274112.6
300000	304593	300988	336834	321408	326970.8

Table 7.7: Hive Execution Time (All time in ms)

Performance of HBase HBase shows similar performance as Hive. Only in the case of two slave nodes, and execution with 1.20 lakh data, a sudden drop in execution time is observed. The results are shown in Table 7.8 and in Figure 7.4.

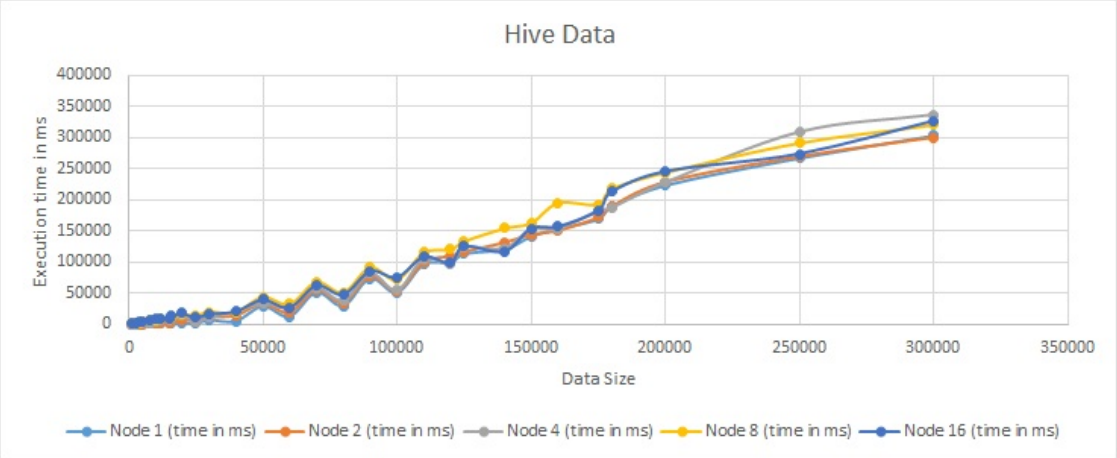

Figure 7.3: Hive execution time 1-16 nodes

Dataset Size	Node 1	Node 2	Node 4	Node 8	Node 16
1000	172	194	543	765	1309
3000	276	378	793	987	1987
4000	341	432	1121	2037	3649
5000	489	458	3107	5101	5800
8000	872	1243	2842	5563	6549
10000	1022	2537	5730	7707	8431
12000	1234	2540	6748	8745	8973
15000	1544	4617	8352	2369	11262
16000	1641	4321	9845	12344	11297
20000	2239	8753	15434	17453	19833
25000	2620	8775	13598	15954	16725
30000	6000	5061	11450	21247	18494
40000	4571	14573	19373	20946	22739
50000	29408	28612	35776	46417	42246
60000	13321	19720	27653	33438	28373
70000	52815	52183	60103	71587	65996
80000	30921	35427	39875	50254	49832
90000	76223	75753	84429	96758	89747
100000	52735	51324	55231	72438	78893
110000	99630	99324	108756	121928	113499
120000	88935	11273	97353	136482	109222
125000	117285	11700	127000	140806	121312
140000	132093	147352	132218	163898	129639
150000	146445	146465	157508	172269	161001
160000	167405	167397	163883	207464	167302
175000	175704	175929	187816	203732	190690
180000	192841	193642	185463	226483	228366
200000	234372	234659	238454	257997	253773
250000	257351	284566	319567	308433	276934
300000	331291	329472	339894	338785	337483

Table 7.8: HBase Execution Time (All time in ms)

Performance of MongoDB In the case of MongoDB, it is noticed that when the node count is increased, the processing time decreases as shown in Table 7.9 and in Figure 7.5.

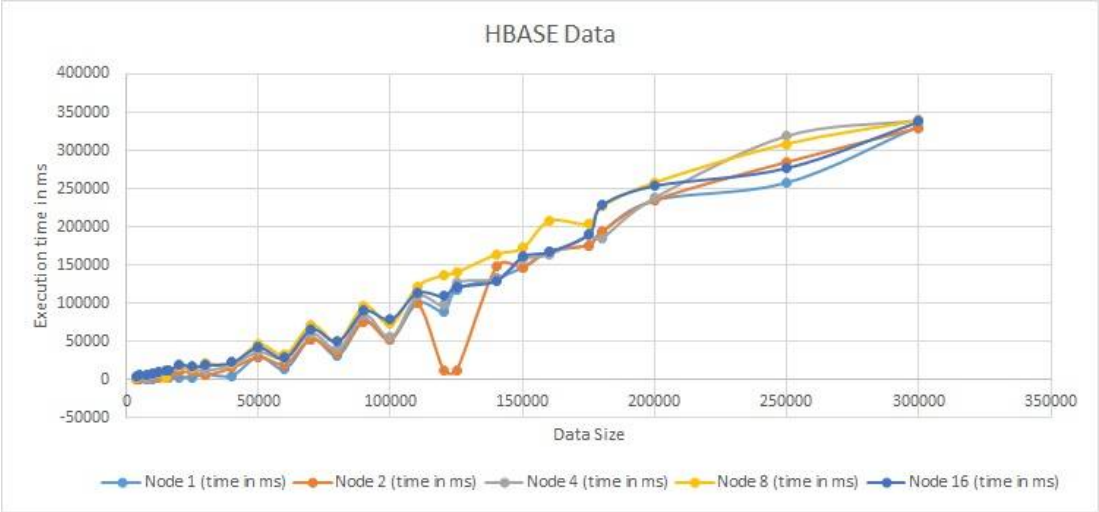

Figure 7.4: HBase execution time 1-16 nodes

Dataset Size	Node 1	Node 2	Node 4	Node 8	Node 16
1000	73.4	56.38	52.98	49.93	45.56
3000	179.86	140.34	130.81	110.2	103.6
4000	346.45	233.79	39.96	184.13	119.4
5000	287.6	237.7	196.21	68.12	178.78
8000	552.93	428.61	234.11	353.34	158.85
10000	563	453.8	338.67	295.15	269.94
12000	759.42	623.43	428.26	522.55	310.49
15000	914.27	769.55	573.87	649.46	424.2
16000	965.89	818.25	622.41	691.77	462.12
20000	1102.26	1017.8	823.58	749.22	652.24
25000	1430.47	1256.59	1059.24	1072.49	803.29
30000	1688.57	1500.12	1301.93	1284.01	992.84
40000	2455.72	1924.2	1793.41	1657.36	1416.85
50000	2720.97	2474.21	2272.68	2130.08	1751.1
60000	3279.18	2821.8	2763.24	2565.5	2181.47
70000	3753.36	3448.3	3243.43	2976.14	2409.17
80000	4401.6	3883.4	3733.07	3473.64	2946.08
90000	4785.76	4422.4	4214.17	3822.2	3267.34
100000	5187.8	5159.2	4702.9	4381.78	3710.69
110000	5907.78	5479.8	5187.82	4835.85	4093
120000	6448.37	5976.53	5672.73	5289.92	4475.3
125000	6592.46	6127.06	5912.98	5302.82	4594.14
140000	7512.5	6982.57	6642.57	6198.06	5239.91
150000	7882.95	7322.68	7126.41	6360.4	5541.85
160000	8564.87	7982.3	7612.4	7106.2	5006.52
175000	9173.45	8562	8339.85	7417.98	6498.56
180000	9528.85	8987.9	8582.23	8014.33	6769.14
200000	10689.96	9983.46	8745.3	7754.9	6656.6
250000	12021.89	11234.77	10987.1	9963.5	9163.32
300000	16006.73	14986.39	15761.8	12997.4	12175.1

Table 7.9: MondoDB Execution Time (All time in ms)

Performance of Cassandra On the other hand, in the case of Cassandra, it is observed that by increasing the node count by fixing the data size, the processing time is increased. The results are shown in Table 7.10 and graph in Figure 7.6. The poor performance on a larger number of slave nodes may be caused due to improper selection of the partition key.

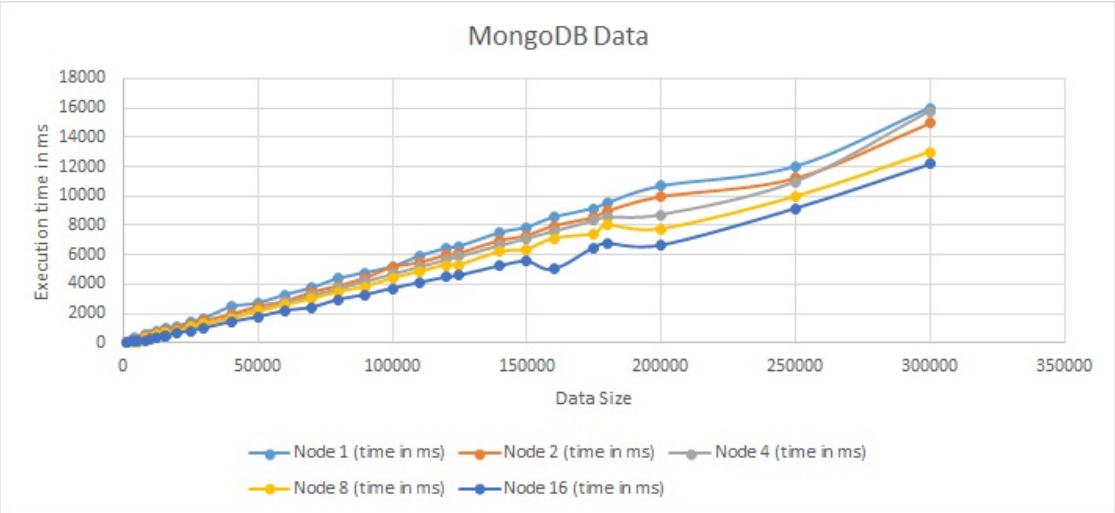

Figure 7.5: MongoDB execution time 1-16 nodes

Dataset Size	Node 1	Node 2	Node 4	Node 8	Node 16
1000	8.9	20	20	24	32.75
3000	22.9	47.85	40.6	61	69.13
4000	24.29	50.98	44.8	65.5	74.9
5000	20	18	12	54	41.76
8000	29.88	63.49	58.53	80.4	98.2
10000	33.3	29	34	91	55.65
12000	35.46	75.99	72.8	95.44	121.52
15000	45.5	54	67	122	109.68
16000	41.05	88.04	87.07	110.2	144.8
20000	51.47	68	95.4	138	156.17
25000	54.8	130	136	147	222.63
30000	66.3	173	149	178	243.9
40000	82.43	165	193.7	221	317.14
50000	89.89	193	231	241	379.08
60000	102.9	252	264	276	432.17
70000	111.15	269	280.3	298	458.85
80000	130.87	279	298.3	351	487.62
90000	139.2	334	354	375	578.23
100000	148.8	378	381	399	623.69
110000	172.31	382.37	422.42	462.17	692
120000	186.28	413.68	458.13	499.6	750.21
125000	171.2	391	410	459	671.17
140000	214.2	476.2	529.4	574.4	866.7
150000	230.8	501	587	619	960.9
160000	242.13	538.4	600.1	649.32	983.06
175000	275.64	687	701	739	1147.53
180000	270.06	601.2	672.2	724.1	1099.98
200000	333.08	788	874	893	1430.73
225000	332.9	741.9	832.75	892.58	1361
300000	418.13	836	998	1121	1631.7

Table 7.10: Cassandra Execution Time (All time in ms)

While executing the queries in Cassandra, proper index key/partition key must be chosen to improve the execution performance [83].

Performance of NGMR During the experiment with Big Data storage, it is noticed that the execution time for MapReduce code increases drastically when data size is increased (e.g. in the case of Hadoop). The reason behind this behaviour is that Hadoop runs its MapReduce code on all nodes. But in the case of NGMR, MapReduce code is run on a small set of Datanodes. So execution time is much less than in the previous case. The result is shown in Table 7.11 and in Figure 7.7.

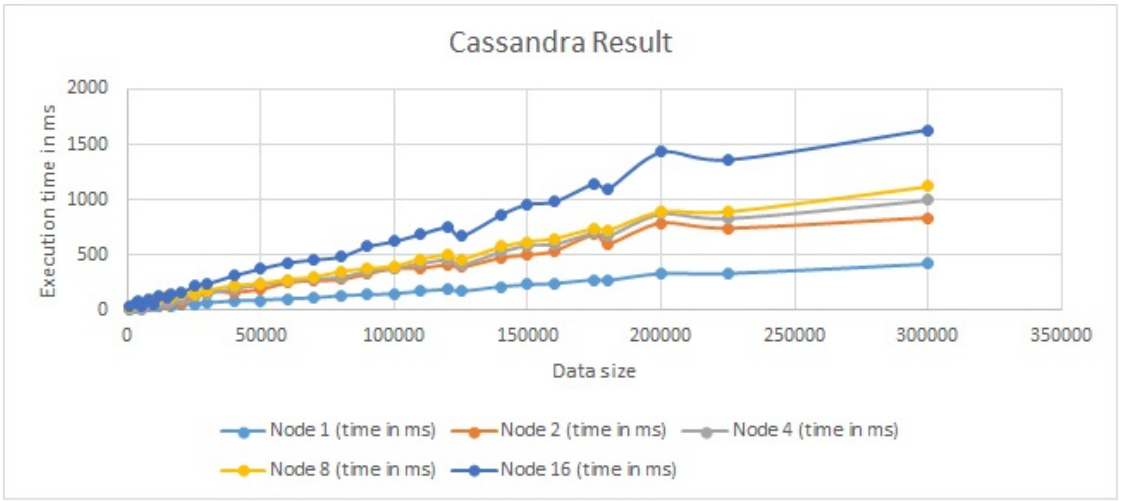

Figure 7.6: Cassandra execution time 1-16 nodes

Dataset Size	Node 1	Node 2	Node 4	Node 8	Node 16
1000	1.3	3.1	6.2	6.1	1.5
3000	2.5	4.2	7.3	7.2	2.7
4000	7.185	7.998	8.087	7.901	8.089
5000	3.64	5.3	8.4	8.3	3.8
8000	7.49	9.108	9.05	9.437	9.006
10000	6.5	8.01	11.19	11.05	6.6
12000	8.288	10.251	9.222	10.202	10.104
15000	9.36	10.82	13.94	13.8	9.4
16000	12.492	11.052	11.257	11.616	11.356
20000	16.63	15.45	13.212	14.796	12.785
25000	15.07	16.26	19.46	19.3	15.1
30000	17.93	18.99	22.21	22.06	17.92
40000	20.542	20.2	22.825	22.68	21.463
50000	29.36	29.89	33.24	33.06	29.19
60000	31.587	33.891	38.081	34.209	30.574
70000	40.79	40.8	44.26	44.07	40.46
80000	38.09	40.63	42.72	45.149	41.535
90000	52.22	51.7	55.28	55.07	51.72
100000	51.349	50.525	68.726	64.451	53.044
110000	63.65	62.6	66.31	66.08	62.99
120000	64.917	62.45	65.98	66.12	62.329
125000	72.23	70.78	74.58	74.33	71.44
140000	78.36	79.893	111.937	108.153	73.72
150000	86.5	84.4	88.36	88.08	85.53
160000	96.056	95.049	98.587	100.31	97.429
175000	100.8	98.04	102.14	101.84	99.62
180000	108.23	109.23	107.2	106.65	109.679
200000	123.7	120.39	118.01	118.79	115.503
250000	145.3	138.03	137.2	140	142
300000	168.3	159.4	160.35	158.3	169.9

Table 7.11: NGMR Execution Time (All time in ms)

In case of the popular programming model MapReduce, the work is carried out in two phases. At first, the application code is distributed throughout the cluster nodes/ Datanodes, which is called the Map phase. In the 2nd phase, after the execution of the application code, the outcome data from all Datanodes is reduced to the Master node as a result. Thus, in traditional MapReduce, the analytical code travels from the Master node to all the Datanodes for execution on the stored data. But sometimes that code may run on some Datanodes where no relevant data is

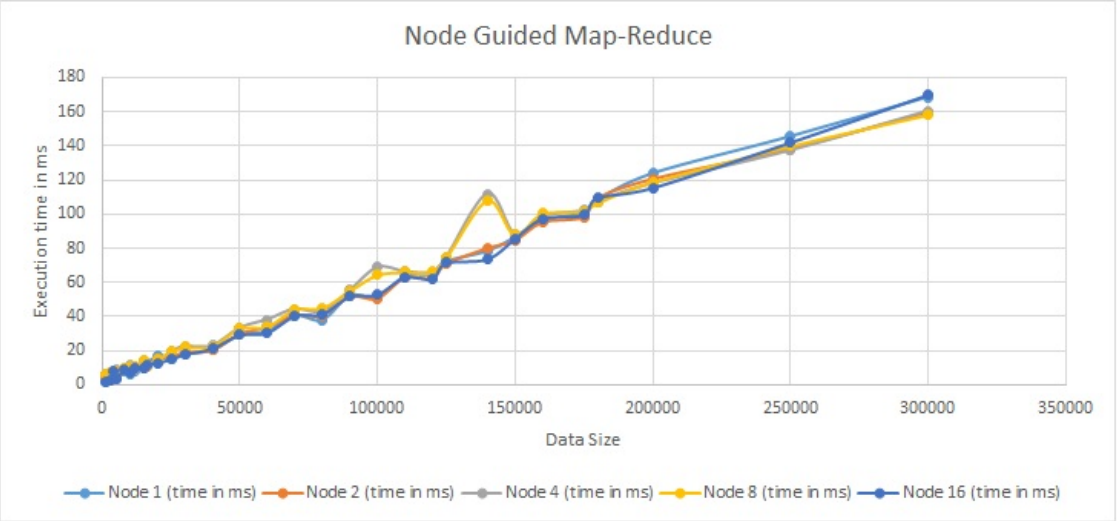

Figure 7.7: NGMR execution time 1-16 nodes

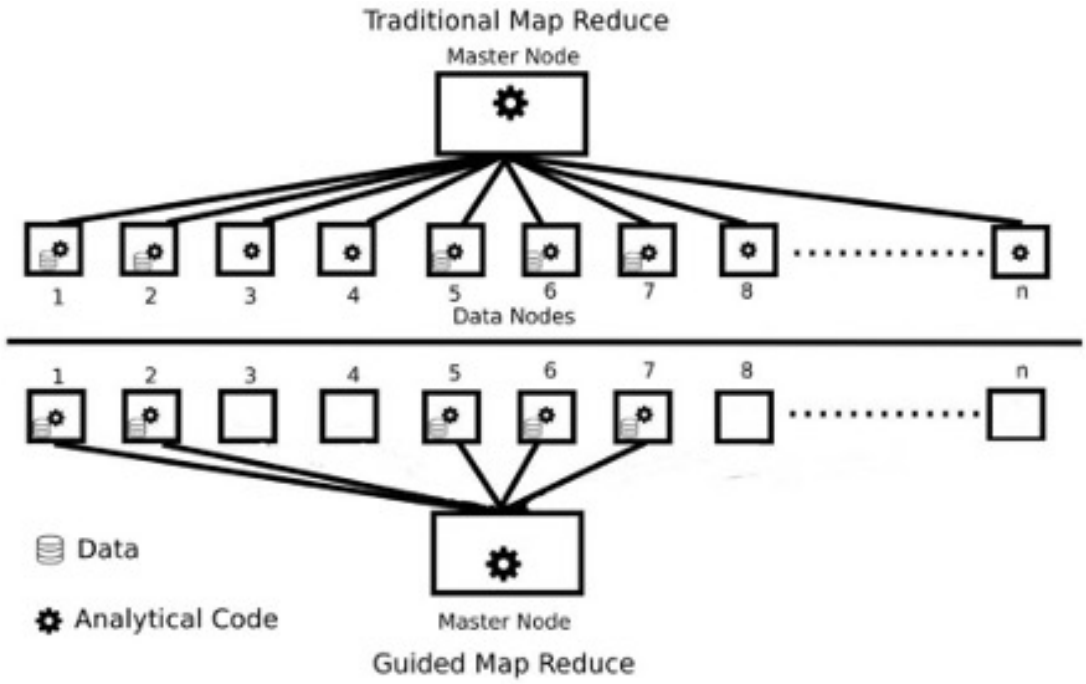

Figure 7.8: Traditional vs. Guided MapReduce

available. For such unnecessary execution, the entire execution time is increased.

But in the case of Node-Guided MapReduce, the Map phase is executed only on those nodes where the required data is stored. This way, unnecessary execution is not initiated and the total execution time is decreased. Figure 7.8 illustrates the working processes in the traditional and guided MapReduce model. If there

are 'n' number of nodes, and if only node 1, node 2, node 5, node 6, and node 7 have the required data, then in the traditional model the code will execute on all nodes; but in the case of guided MapReduce, only node 1, 2, 5, 6 and 7 are the active nodes where MapReduce program will run.

However, in the case of NGMR as well, there is a sudden rise in the execution time, which may be due to the network latency, when the data size is 1.4 Lakh and the node count is 4 and 8.

7.4.2 Comparison of the Big Data solutions

Now, in the next part of the experiment, the number of nodes is varied with a fixed data count. The node count has been taken as 1, 2, 4, 8, and 16. The results are discussed in the following part of the section.

- **Execution on a single node:** In this experiment, the execution time has been measured for data size 1000 to 3 Lakh. The maximum time is taken by Hadoop and the minimum time is taken by NGMR. From Figure 7.9 it is clear that for 3000 data Hadoop completes the same query execution in 111310 sec and NGMR takes only 2.5 sec. The other tools are Hive, HBase, MongoDB and Cassandra take 376sec, 276sec, 180 sec, and 22.9 sec respectively.

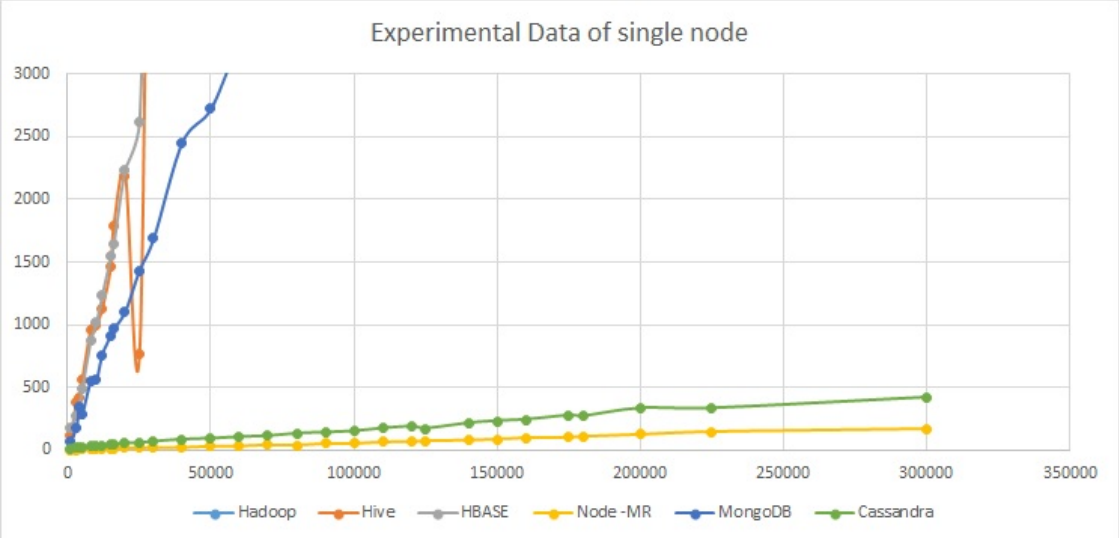

Figure 7.9: Experimental data with various tools in single node

- **Execution on 2 nodes:** In the case of a two-node cluster, the minimum or less time is taken for the same query execution by NGMR. The next minimum time is taken by Cassandra, and then MongoDB. HBase and Hive take more or less the same time. Maximum time is taken by Hadoop. Figure 7.10 shows the result.

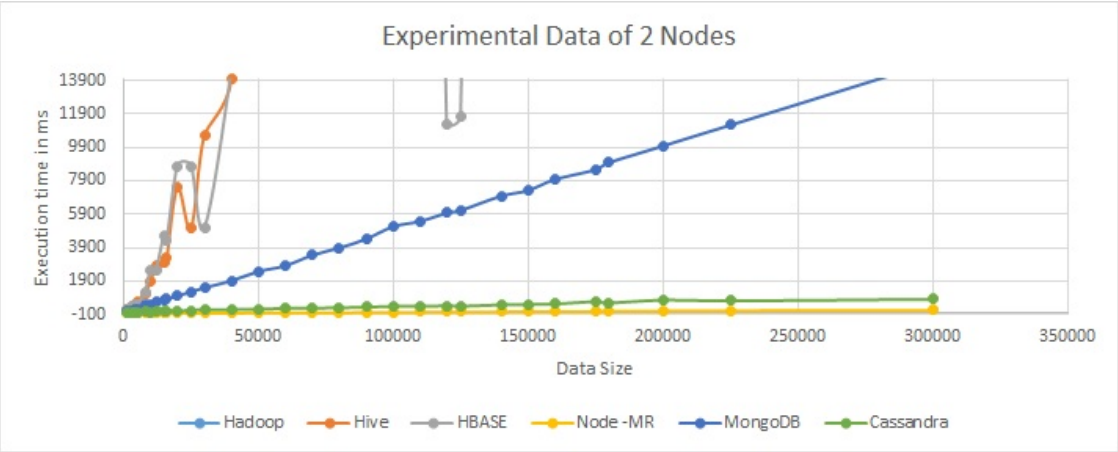

Figure 7.10: Experimental data with various tools in two nodes

- **Execution on 4 nodes:** In the case of the experiment with 4 nodes, the results are similar to the experiment on 2 nodes. Here as well minimum time is taken by NGMR and maximum time by Hadoop. Figure 7.10 shows the result.

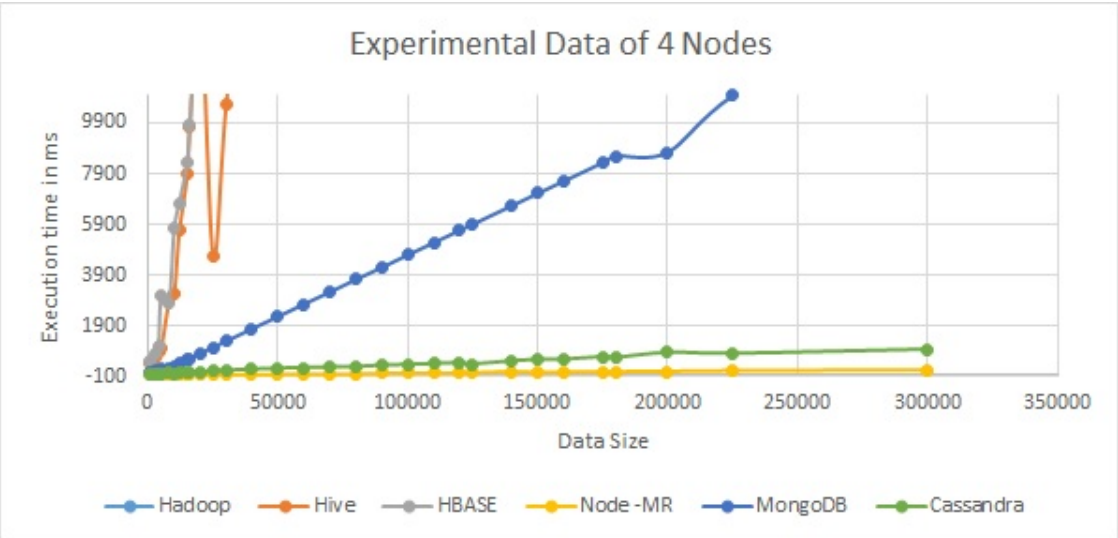

Figure 7.11: Experimental data with various tools in four nodes

- **Execution on 8 nodes:** The experiment with 8 nodes returns the result like the previous two cases. If the tools are sorted in ascending order depending on the query execution time, then it will be NGMR, Cassandra, MongoDB, Hive, HBase, and at last Hadoop. Figure 7.10 shows the result.

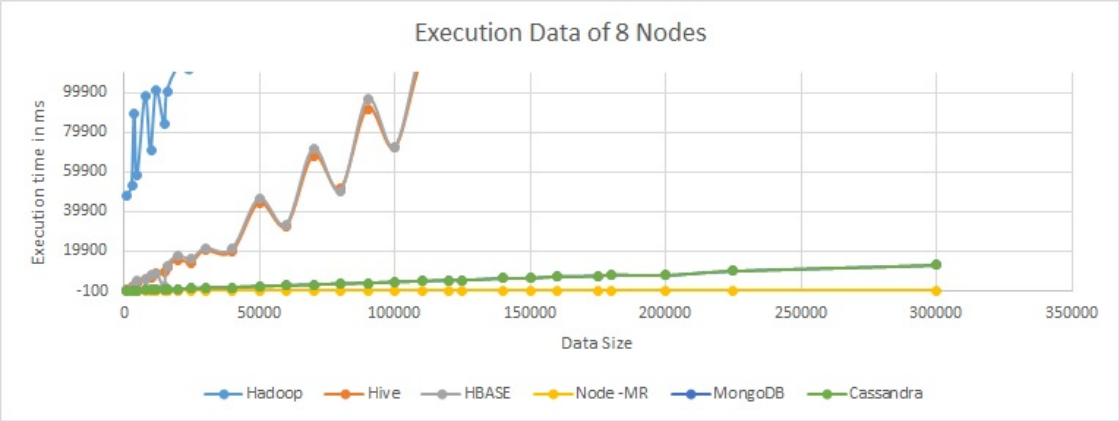

Figure 7.12: Experimental data with various tools in eight nodes

- **Execution on 16 nodes:** At the time of the experiment with 16 modes, it is observed that the results were the same as in the previous cases. Only the execution time is increased as there are more nodes associated with the master node. From the experimental data, it is shown that the processing time is around 90 sec for only 1000 data in 16 nodes in the case of Hadoop. In contrast, it takes only 1.5 ms, 32 ms and 42 ms in the cases of NGMR, Cassandra and MongoDB respectively. The execution time using 3 Lakh data, in the case of then NGMR is only 170ms to execute the same query whereas Cassandra, MongoDB, Hive, HBase and Hadoop take 1.6 Sec, 12.1 Sec, 5.4 Min, 5.6 Min, and 13.3 Min respectively. The obvious reason for the increase in the time is due to the increase in data size. However, for the result set of NGMR, it is observed that the execution time is much less and the rise in execution time from 1,000 data to 3,00,000 data is nearly linear as shown by the yellow line in Figure 7.13. The reason behind the reduced execution time is that NGMR always runs its code on a smaller set of nodes instead of the whole set of nodes leading to less execution time.

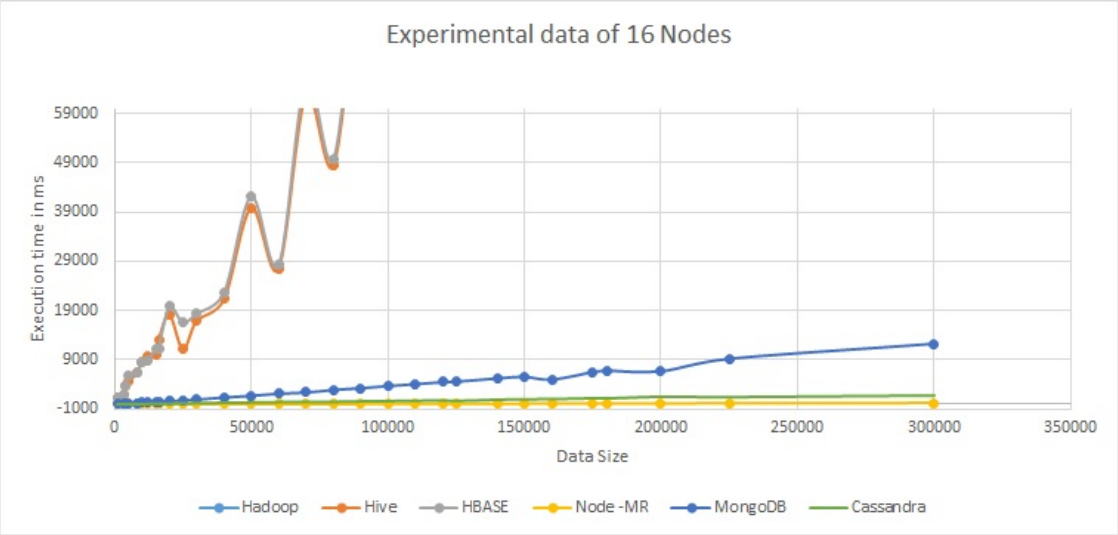

Figure 7.13: Experimental data with various tools in sixteen nodes

The above results also explain that when the processing times are plotted for every tool in a single graph, the performance of NGMR is always better than other storage and tools. From the Figures 7.9, 7.10, 7.11, 7.12, 7.13 it is understandable that Hadoop MapReduce, Hive, HBase give the users flexibility when historical data processing is required. However, for real-time instant data retrieval, the NoSQL databases are always one step ahead of the Hadoop and HDFS-dependent tools.

7.5 Summary

In this chapter, the storage and retrieval of health data in a few Big Data solutions are studied. A healthcare data model proposed based on EHR standards has been considered, and its mappings onto six Big Data solutions—Hadoop, Hive, HBase, Cassandra, and MongoDB—have been demonstrated. A comparative study of query processing performance in the six tools has also been conducted. Conclusions are being drawn regarding the types of tools required for different processes.

It is observed that in case of MongoDB the data was stored in structured form highest throughput value has been observed compared to Cassandra and HBase. Whereas for unstructured data, Cassandra excels at producing the best throughput values [92].

It is also observed that the NGMR always performs better as it stores the data in a Flat file system and maintains the multilevel indexing.

More specific queries can be proposed and tested to store and retrieve of Health Data in the future. Many more tools are available like Apache Pig, Graph Database, and Relational Databases. Multiple research issues can be pointed out with the use of the tools mentioned above for Big Health Data storage. This will be taken up as future work.

Data is a precious thing and will last longer than the systems themselves.

— Sir Tim Berners-Lee

8

Node selection in Node Guided MapReduce

Contents

8.1 Prelude

Node Guided MapReduce is an advanced approach within the realm of distributed computing, specifically tailored to harness the power of large-scale data processing. In Chapter 6 a novel architecture of Node-Guided MapReduce has been discussed. Node-guided MapReduce represents a sophisticated approach to distributed data processing, leveraging node-level optimization techniques to enhance scalability, performance, and resource utilization.

In the MapReduce programming paradigm, massive amounts of data are distributed in parallel storage across a distributed cluster of nodes. But there is no scope to determine, which block is stored on which node. At the time of execution, the two main phases help the execution of the desired queries, one is the Map phase, where data is broken down into key-value pairs and processed in parallel, and the Reduce phase, where the intermediate results from the map phase are aggregated to produce the final output. However, due to non-selective nodes being involved in MapReduce, execution time cannot be optimised. Node-guided MapReduce takes this a step further by optimizing resource utilization at the node level. In Chapter 6, in case of the Node-Guided MapReduce process, the user or the admin manually determined the executable nodes. In this chapter, an approach to automate the selection of the executable nodes is proposed.

8.2 Overview of NGMR

The huge amount of health data cannot be stored in a single conventional storage as the size of the Big Data file is too large to fit in a single system. It should be distributed for fast processing of user-given queries. For handling the Big Data, large data centres are required along with large capacity servers to build the infrastructure. For NGMR node selection algorithm, a cluster with multiple data nodes is used with one master node having higher computation configuration. For the algorithm, it is assumed that there are n number of data nodes, so the set of the nodes N is

$$N = \{n_1, n_2, n_3, ..., n_n\}$$

The collection of data nodes containing all the fragments based on one key attribute is termed *zone*. The details are described in Section 6.4.3. According to Equation 6.2, no *zone* stores two different key-based fragments. Every *zone* must contain all the fragments based on the same key so that even if a *Unit / Node* of a particular *zone* fails, all the fragments of any of the remaining *zones* are accumulated to form the original Big Data file and the particular key-based fragments of the failed node are regenerated by the algorithm itself ensuring failure recovery without data loss. Suppose the set of the zones is Z, then

$$Z = \{z_1, z_2, z_3, ..., z_n\}$$

So, as per the Equation 6.2, the following zones may be considered:

$$z_1 = \{n_1, n_2\}$$

$$z_2 = \{n_{10}, n_3\}$$

$$z_3 = \{n_{11}, n_5, n_{12}, n_9\}$$

$$z_4 = \{n_8, n_4, n_7, n_{13}\}$$

Thus, if the data of $[n_1]$ and $[n_2]$, or the data of $[n_{10}]$ and $[n_3]$ are combined the original file with all records can be reconstructed.

The data is stored based on the keys, The data must have the set of keys K so that

$$K = \{k_1, k_2, k_3, ..., k_m\}$$

The data is clustered by keys at the time of storage. For example, data associated with the key $[k_1]$ is stored across the nodes $[n_1, n_2]$, while data associated with the key $[k_2]$ is stored across the nodes $[n_{10}, n_3]$. From another perspective, nodes $[n_1, n_2]$ form zone $[z_1]$, and nodes $[n_{10}, n_3]$ form zone $[z_2]$. Therefore, it can be clearly stated that $[k_1]$ belongs to zone $[z_1]$, and $[k_2]$ belongs to zone $[z_2]$.

The following subsections presents an approach to find the best zone $[z_y]$ for a given query.

8.2.1 Computation of various factors

To calculate the best set of nodes for processing a query, some factors related to the nodes are computed. The factors are:

- CPU Factor
- Memory Factor
- Storage Factor
- Health Factor
- Bandwidth Factor
- Latency Factor

Each factor has a unique method or formula to determine its value for the Qn query, resulting in a different value for each query.

- **CPU Factor**: Determining the CPU factor is needed to know the CPU performance which often depends on the specific context. However, a common approach is to use the CPU usage percentage as a metric for overall CPU performance [68].

  An inverse relationship is assumed between CPU usage percentage and performance, where lower CPU usage is considered healthier. The formula for calculating CPU factor based on CPU usage percentage can be:

  CPU performance (CP_i) **= 1 - (CPU usage percentage during the execution time of the query** Q_j **/ 100)**

  Min CP_i **= Min CPU performance value in time** t
  Max CP_i **= Max CPU performance value in time** t

  CPU Factor (CF_i) **= (** CP_i **- Min** CP_i**) / (Max** CP_i **- Min** CP_i **)**

  CPU performance is the actual CPU performance of the node N_i. Min CPU performance is the minimum CPU performance of node N_i in the cluster.

Max CPU performance is the maximum CPU performance of node N_i in the cluster.

- **Memory Factor**: Execution of the query Q_j on node N_i for the duration of time t, then the memory factor of node N_i is

  MF_i = **(Free memory size during execution of** Q_j **/t) / Total memory size in Node** N_i

- **Storage Factor**: Storage factor calculation is the same as how memory factor is calculated. The storage factor for query Q_i is

  SF_i = **(Free storage size during execution of** Q_j **/ t) / Total available storage size of the node** N_i

- **Health Factor** : Health factor is calculated by the health of the node N_i. The health is calculated by the completed job/task submitted to the node N_i

  HF_i = **query execution completed successfully in time t / Total query executed in time** t

- **Bandwidth Factor**: Network Bandwidth between the node and the master node is calculated based on the maximum bandwidth between the master node and a particular node N_i. So bandwidth factor of N_i is

  BF_i = **Available bandwidth between master-node and** N_i **at the time of execution of** Q_j **/ Maximum calculated bandwidth between master-node and** N_i

- **Latency Factor**; CPU latency is the time delay or the amount of time it takes for the CPU to complete a specific operation or task. It is a measure of the time it takes for the CPU to retrieve data from memory, perform calculations, and generate the desired output. To calculate the latency factor for a node N_i to get the clock cycle of IO for a query, and then the waiting time for a given query.

  IO time = Clock cycles for an IO for a query / Clock rate

Wait time = Clock cycles for waiting to run a query / Clock rate

Current Latency = (IO time + wait time) for query Q_j

Maximum Latency = maximum calculated (IO time + wait time)

LF_i **= Current latency / Maximum Latency**

8.2.2 Weight assignment

Weights are assigned to each factor based on their relative importance in determining the health of a node. For example, if CPU performance is deemed to be more critical for the application, a higher weight can be assigned to the CPU factor compared to other factors. The following weights may be assigned between 0 and 1.

$$CPU => 0.2,$$
$$Memory => 0.15,$$
$$Storage => 0.1,$$
$$Health => 0.25,$$
$$Bandwidth => 0.15,$$
$$Latency => 0.15,$$

In this chapter, it is assumed that all the factors have the same importance to calculate the healthier node for a given query execution. So, the same value, 1 has been assigned to all weights of the factor.

$$CPU => 1/6,$$
$$Memory => 1/6,$$
$$Storage => 1/6,$$
$$Health => 1/6,$$
$$Bandwidth => 1/6,$$
$$Latency => 1/6,$$

8.2.3 Determination of query keys

First it is required to determine the keys from a query. Extraction of keys from a query is a crucial step. Based on this step, the algorithm detects the nodes which are needed to execute.

8.2.4 Determination of node stored-keys

For each node, a record of stored keys is maintained. This means that a metadata file containing the key values is stored, which is then used to store the data. The metadata is stored in both the master node and the data node. The master node holds information about the data node, while the data node stores information about the data blocks.

8.2.5 Normalisation of factors

Normalisation is required for each factor to bring them to a common scale. This step is essential to ensure that each factor contributes proportionally to the overall health assessment. Various normalisation techniques, such as Min-Max scaling or Z-score normalisation, can be used. Simple Min-Max scaling has been applied.

8.2.6 Score calculation

A health score for each node is computed by combining the normalised factors with their assigned weights. A weighted sum formula is used, where the health score (H) for each node is calculated as follows.

$$\mathbf{H} = w_1 \ \mathbf{X} \ f_1 + w_2 \ \mathbf{X} \ f_2 + \ldots . + w_n \ \mathbf{X} \ f_n$$

8.2.7 Rank calculation

The nodes are ranked based on their calculated health scores. Only the selected nodes that contain the respective keys are calculated by the application. The node with the highest health score is considered the healthiest. The set of healthier nodes for a particular query can also be calculated.

8.2.8 Finding healthy node by historical rank

The historical performance of the nodes are also important, thus a running average or some other metric over a window of 7 days may be considered to check the health of the nodes.

8.3 Explanation of the algorithm with example

In this section, the algorithm is discussed along with an example. Let us assume

Set of nodes, N = $n_1, n_2, n_3, ..., n_8$

Set of zones, Z = $z_1, z_2, z_3, ..., z_8$, such that

$$z_1 = \{n_1, n_2\}$$

$$z_2 = \{n_{10}, n_3\}$$

$$z_3 = \{n_{11}, n_5, n_{12}, n_9\}$$

$$z_4 = \{n_8, n_4, n_7, n_{13}\}$$

The data is distributed and stored based on the keys of the data.

A set of keys, K = $k_1, k_2, k_3, ..., k_m$ is assumed, where

$$k_1 => Age$$

$$k_2 => Disease$$

$$k_3 => FoodHabit$$

$$k_4 => City$$

$$k_5 => Occupation$$

As $K_x \in Z_y$, Where $1 \leq x, y \leq n$, assuming that,

$$k_1 \in z_4,\ so\ z_4 => Age$$

$$k_2 \in z_1,\ so\ z_1 => Disease$$

$$k_3 \in z_2,\ so\ z_2 => Foodhabit$$

$$k_4 \in z_3,\ so\ z_3 => City$$

That means, in zone z_1 data is distributed using the data key disease, The nodes are filled with the data, with the keys of various disease names. *Zone* z_1 stores all the data based on the key "Disease", the nodes of the *Zone* (n_1, n_2, n_3) should have all the data based on the Key "Disease". The large Big Data file can be retrieved by combining the stored data of (n_1, n_2, n_3).

Next the keys in the query are extracted. The following query is considerd,
Q1: Get all details of the patients who are suffering from diabetes, age group between 40 to 50 and belong to Kolkata and Mumbai
From the above query the extracted keys are ,

1. Disease (Diabetes)
2. Age (40> age <50)
3. City (Kolkata and Mumbai)

If all the nodes are working and any of the nodes are not down, then any one of the 3 zones can be selected to execute the above query:

1. z_1 for Disease
2. z_4 for Age and
3. z_3 for City.

Suppose,

- The required data for key-value 'Diabetes' is stored in { n_1, n_2 } in z_1 with 'Disease' as key.
- The required data for key-value '40> age <50' is stored in { n_4, n_7, n_{13} } in z_4 with 'Age' as key.
- The required data for key-value 'Kolkata and Mumbai' are stored in { n_{11}, n_5, n_9 } in z_3 with 'City' as key

For the above Query Q1, it is necessary to find which of the set of nodes is healthier and which set of nodes is best to execute the above query.

The pseudo code of the Algorithm 6 describes the process of normalising node data, calculating health scores, implementing a sliding window for averaging, and identifying the healthiest node.

8.3.1 Simulation of the algorithm

Due to the shortage of the resources, only one virtual machine in AWS [88] has been created. The VM has 80 GB of HDD space with 2 core processor with 8 GB of RAM. Figures 8.1, 8.2, 8.3 show the details of the configurations.

```
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1020-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

  System information as of Mon May 20 21:07:12 UTC 2024

  System load:                      1.01806640625
  Usage of /:                       94.5% of 77.35GB
  Memory usage:                     54%
  Swap usage:                       0%
  Processes:                        167
  Users logged in:                  0
  IPv4 address for br-b00ecfaaaa4f: 172.22.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            10.200.12.130

  => / is using 94.5% of 77.35GB

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

10 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


Last login: Sat May 18 06:04:58 2024 from 45.250.246.25
```

Figure 8.1: AWS virtual machine system information

The node selection code is running in a separate engine, the code is written in PHP - Laravel framework. This code produces an output as an associative array

Algorithm 6 Algorithm for Node selection

Initialize factors ← {CPU = 1.0, Memory = 1.0, Storage = 1.0, Health = 1.0, Bandwidth = 1.0, Latency = 1.0}
Initialize nodeData ← {Node5: {CPU: 90, Memory: 80, Storage: 95, Health: 85, Bandwidth: 70, Latency: 75},
Node4: {CPU: 85, Memory: 75, Storage: 30, Health: 44, Bandwidth: 82, Latency: 82},
Node3: {CPU: 35, Memory: 73, Storage: 50, Health: 40, Bandwidth: 84, Latency: 52},
Node2: {CPU: 65, Memory: 71, Storage: 63, Health: 54, Bandwidth: 30, Latency: 34},
Node1: {CPU: 56, Memory: 35, Storage: 50, Health: 91, Bandwidth: 43, Latency: 57}}
Initialize normalizeNodeData ← {}
for all factor **in** factors **do**
    minVal ← min(array_column(nodeData, factor))
    maxVal ← max(array_column(nodeData, factor))
    for all (key, val) **in** nodeData **do**
        temp ← $\frac{(val[factor]-minVal)}{(maxVal-minVal)}$
        temp ← number_format(temp, 3, ".", "")
        normalizeNodeData[key][factor] ← temp
    end for
end for
Initialize healthScores ← {}
for all (key, val) **in** normalizeNodeData **do**
    sum ← 0.0
    for all (fkey, fval) **in** factors **do**
        sum ← sum + (val[fkey] * fval)
    end for
    healthScores[key] ← sum
end for
Initialize windowSize ← 7
Initialize windowHealthScores ← {}
Initialize healthiestNodes ← {}
for all (node, score) **in** healthScores **do**
    append(windowHealthScores, score)
    if (length(windowHealthScores) > windowSize) **then**
        array_shift(windowHealthScores)
    end if
    if (length(windowHealthScores) = windowSize) **then**
        averageHealthScore ← $\frac{\text{sum}(windowHealthScores)}{\text{length}(windowHealthScores)}$
        healthiestNodes[node] ← averageHealthScore
    end if
end for
arsort(healthiestNodes)
healthiestNode ← key($healthiestNodes$[0])
print("Healthiest Node (based on sliding window average): ", healthiestNode)

```
ubuntu@ip-10-200-12-130:~$ lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         46 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  2
  On-line CPU(s) list:   0,1
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
    CPU family:          6
    Model:               79
    Thread(s) per core:  1
    Core(s) per socket:  2
    Socket(s):           1
    Stepping:            1
    BogoMIPS:            4600.01
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
                          cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx
                         rdtscp lm constant_tsc rep_good nopl xtopology cpuid tsc
                         _known_freq pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse
                         4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx
                          f16c rdrand hypervisor lahf_lm abm cpuid_fault invpcid_
                         single pti fsgsbase bmi1 avx2 smep bmi2 erms invpcid xsa
                         veopt
Virtualization features:
  Hypervisor vendor:     Xen
  Virtualization type:   full
Caches (sum of all):
  L1d:                   64 KiB (2 instances)
  L1i:                   64 KiB (2 instances)
  L2:                    512 KiB (2 instances)
  L3:                    45 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0,1
```

Figure 8.2: AWS virtual machine processor information

```
ubuntu@ip-10-200-12-130:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           7.7Gi       1.8Gi       497Mi       150Mi       5.4Gi       5.5Gi
Swap:             0B          0B          0B
ubuntu@ip-10-200-12-130:~$
```

Figure 8.3: AWS virtual machine RAM information

with key as node name and the weight as the value.

Due to resource limitations, multiple microservices are deployed as a collection of independent services. Each service is treated as an independent virtual machine. The microservices are executed with variable and random dynamic task loads to simulate the characteristics of real-time independent cluster nodes.

Figure 8.4 shows the output of the node selection engine. This figure is an output screenshot taken after execution of the query Q1, The query is executed

more than 20 times and average of the execution time is taken.

```
Array
(
    [Node5] => Array
        (
            [CPU] => 1.000
            [Memory] => 1.000
            [Storage] => 1.000
            [Health] => 0.882
            [Bandwidth] => 0.741
            [Latency] => 0.854
        )

    [Node4] => Array
        (
            [CPU] => 0.909
            [Memory] => 0.889
            [Storage] => 0.000
            [Health] => 0.078
            [Bandwidth] => 0.963
            [Latency] => 1.000
        )

    [Node3] => Array
        (
            [CPU] => 0.000
            [Memory] => 0.844
            [Storage] => 0.308
            [Health] => 0.000
            [Bandwidth] => 1.000
            [Latency] => 0.375
        )

    [Node2] => Array
        (
            [CPU] => 0.545
            [Memory] => 0.800
            [Storage] => 0.508
            [Health] => 0.275
            [Bandwidth] => 0.000
            [Latency] => 0.000
        )

    [Node1] => Array
        (
            [CPU] => 0.382
            [Memory] => 0.000
            [Storage] => 0.308
            [Health] => 1.000
            [Bandwidth] => 0.241
            [Latency] => 0.479
        )

)
Array
(
    [Node5] => 5.477
    [Node4] => 3.839
    [Node3] => 2.527
    [Node2] => 2.128
    [Node1] => 2.41
)
Healthiest Node: Node5
```

Figure 8.4: Output of the node selection engine

8.4 Query

There are nine queries presented in Section 7.3.2 which have been used in the framework. From those nine queries, the following query is used to explain the procedure of node selection.

- Q_1: Get all details of the patients who are suffering from diabetes, age group between 40 to 50 and belong to Kolkata and Mumbai

8.5 Experiment, result and discussion

For experiment purpose, the same dataset described in Section 7.2, consisting of more than 8 lakh records populated synthetically, has been used.

8.5.1 Uploading

The data is stored in the cluster in two ways: first, the data is divided into blocks and stored in the cluster nodes. Next, the data is divided into key-value pairs, formed into blocks, and then stored in the cluster nodes. In both cases, the upload and storage times are recorded. The recorded values are displayed in the Tables 8.1, 8.2. Figures 8.5, 8.6, 8.7 illustrate the upload and storage times. For query execution, the same codebase described earlier in Section 6.8.1, built with Node.js [90], is used.

Data count	1 Node	2 Nodes	4 Nodes
10	0.344	0.782	0.963
50	0.541	0.957	1.221
4000	15.324	20.217	23.74
8000	28.91	34.741	75.921
12000	32.194	67.662	83.402
16000	45.534	78.549	101.715
20000	59.677	89.223	123.876
40000	119.823	136.776	157.025
60000	293.121	317.164	386.725

Table 8.1: Data upload time without application of node selection algorithm

The data upload times are displayed in Figures 8.5, 8.6, 8.7 for various data counts across 1, 2, and 4 nodes both with and without the "Node Selection" algorithm.

Data Count	1 Node	2 Nodes	4 Nodes
10	2.5	4.89	5.46
50	4.87	6.36	17.484
4000	31.395	45.43	48.294
8000	54.502	73.51	97.464
12000	67.491	97.33	174.627
16000	153.8	199.95	194.411
20000	194.02	287.64	231.393
40000	305.491	561.02	427.574
60000	478.923	808.95	660.219

Table 8.2: Data upload times with node selection algorithm

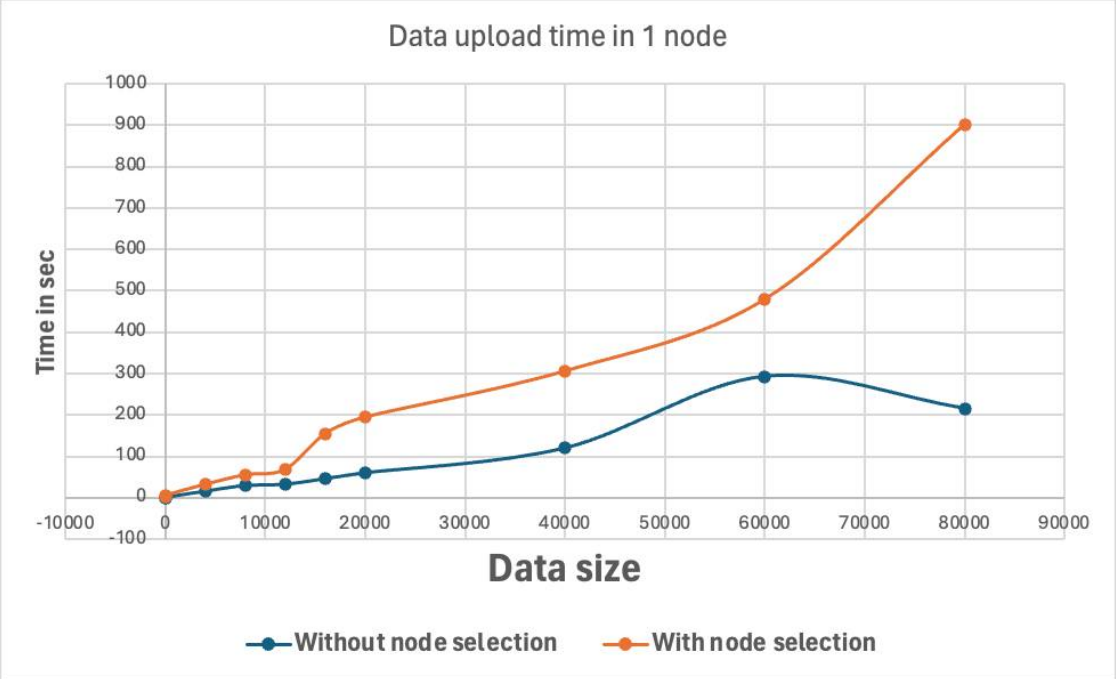

Figure 8.5: Data upload times on one node

Notably, the data shows greater upload times for the scenarios when node selection is used. This indicates that the 'Node Selection' algorithm takes a longer time to store because it divides the data based on the keys, then distributes and stores it on the distributed nodes. Consequently, the upload performance remains higher when the algorithm is employed. It can be inferred that the "Node Selection" algorithm, is best for when writing once and reading multiple times is applied.

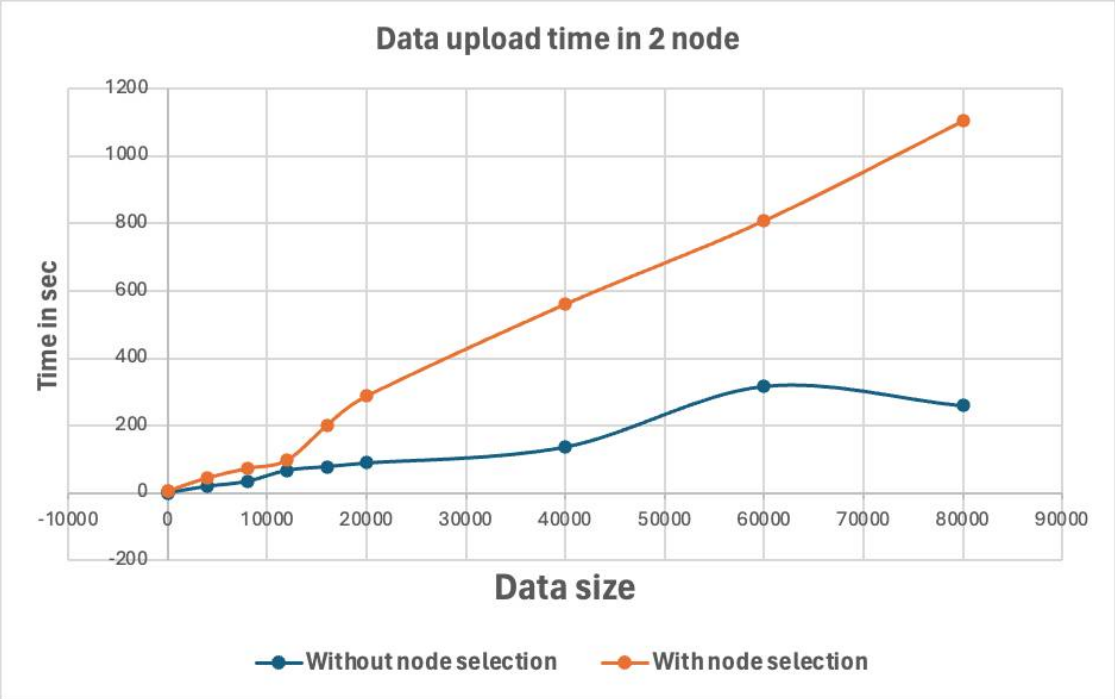

Figure 8.6: Data upload time on two nodes

Figure 8.7: Data upload times on four nodes

8.5.2 Execution

In the case of execution of the query, it is noticeable that the execution time is much less when the node selection algorithm is applied. Figures 8.8, 8.9, 8.10

Data Size	1 Node	2 Nodes	4 Nodes
10	12.7	13.1	13.5
50	12.893	12.91	13.02
4000	16.62	16.71	16.83
8000	30.34	31.04	30.78
12000	50.33	51.01	51.43
16000	72.04	72.37	72.65
20000	95.227	95.381	95.865
40000	145.88	146.03	145.49
60000	286.9	386.94	2867.2

Table 8.3: Execution times when node selection algorithm is not applied

Data Size	1 Node	2 Nodes	4 Nodes
10	3.47	3.6	3.894
50	4.82	5.1	5.26
4000	5.02	5.21	5.34
8000	6.53	6.82	7.01
12000	6.83	6.87	6.91
16000	7.27	7.3	7.36
20000	8.3	8.49	8.53
40000	8.78	8.792	8.81
60000	10.1	10.38	10.44

Table 8.4: Execution time when node selection algo is applied

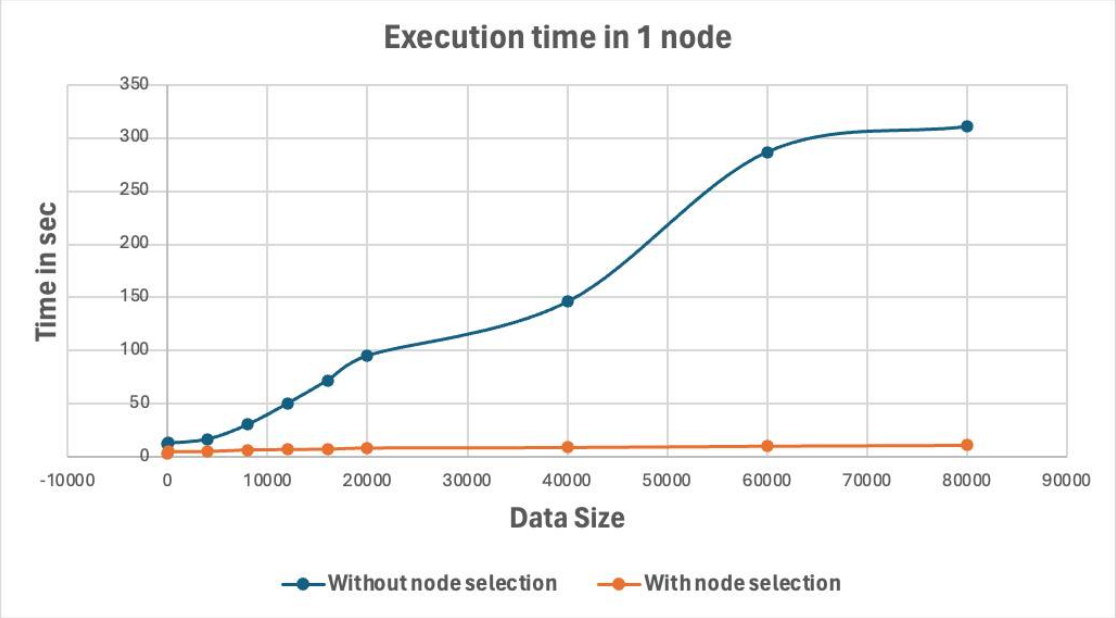

Figure 8.8: Execution time on one node

Figure 8.9: Execution time on two node

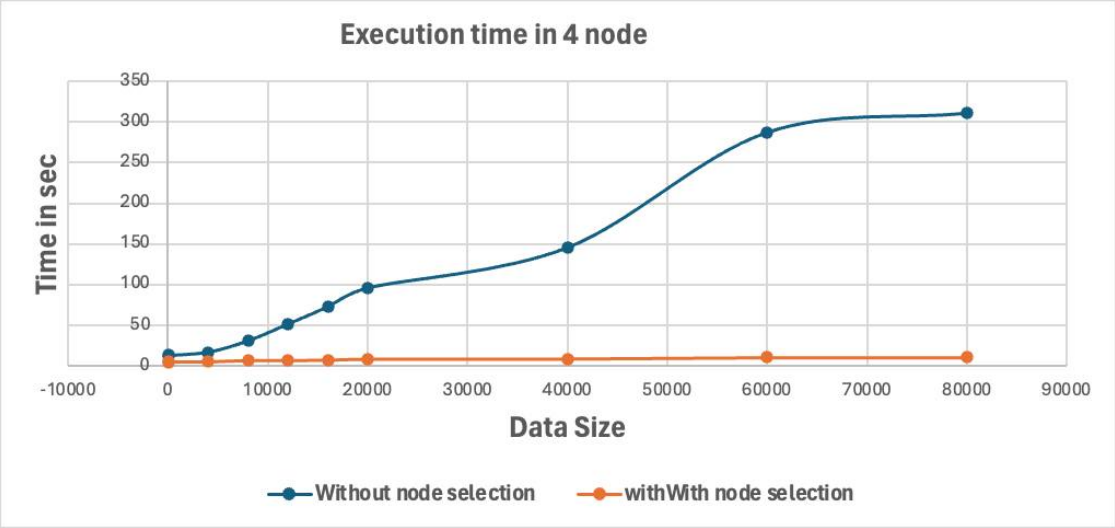

Figure 8.10: Execution time on four node

and Tables 8.4, 8.3 shows that the execution time in node one, two and four respectively. For example, when 4 nodes are available, and data size is 8 lakh, then it takes only 10 seconds to fetch the record from the nodes. On the other hand, more than 300 seconds is required when the query is executed without node selection. TThis is because, without node selection, the query is executed across all four nodes, even if the relevant data isn't found on all of them, leading to a much longer overall execution time.

8.6 Summary

In this chapter, automated node selection algorithm for Node-Guided MapReduce (NGMR) is discussed. Node-Guided MapReduce (NGMR) is an advanced approach for distributed data processing. In traditional MapReduce, the nodes for data processing are not selectively chosen, leading to inefficiencies. NGMR addresses this by introducing node selection, where nodes are chosen based on factors like CPU, memory, storage, health, bandwidth, and latency. A weighted algorithm calculates the health score for each node, allowing more efficient query execution. This process is particularly useful in large-scale systems, like healthcare databases, where data is distributed across multiple zones based on key attributes. The NGMR approach ensures that queries, such as retrieving patient data for specific conditions, are processed using the healthiest nodes, improving both speed and reliability. Additionally, a historical ranking system can further optimize performance by considering nodes' past performance. A testbed is made also made using microservices to implement the node selection algorithm.

The goal is to turn data into information, and information into insight.

— Carly Fiorina, CEO, Hewlett-Packard

9 Conclusion

Contents

9.1 Prelude

The research work begins by identifying challenges in Big Data in remote health-care, particularly in resource-poor environments where doctors, nurses, medicine, electricity, and Internet connectivity are not easily available, infrastructure is not adequate, and a lack of expertise exists, thereby creating obstacles to good services. The thesis highlights the potential of eHealth, mHealth, and digital health to provide patient-centric remote health-care solutions in order to improve health-care systems. The thesis also focuses on the role of Big Data in enhancing storage, efficient retrieval, and better communication.

The thesis focuses on efficient data storage and retrieval, comparing Big Data tools, and introducing Node-guided MapReduce, which optimizes node selec-

tion by automated node selection algorithm to improve resource utilization and execution time,

9.2 Overview of the thesis

This research work is driven by the goal of implementing primary health-care services using Information and Communication Technology (ICT) and cloud collaboration. The challenges associated with Big Data in health-care are detailed in Chapter 1.

Chapter 2 explores the large domains of eHealth, Digital Health, and Mobile Health, which are recent pillars of health-care. eHealth is at the forefront of this revolution, offering a comprehensive, patient-centric approach to medical care. As technology continues to advance, integrating eHealth solutions will be crucial in shaping the future of health-care delivery. By addressing key challenges and making collaboration between patients, health-care providers, policymakers, and software developers, the full potential of eHealth can be realized. This will make a health-care system efficient, accessible, and personalized for every individual.

The primary aim of this research work, explored in Chapter 3, is delivering remote health-care services to rural people. The integration of eHealth and mHealth through Information and Communication Technology (ICT) with health-care services, particularly in rural areas has been taken up in this chapter. While eHealth leverages Electronic Health Records (EHRs) and telemedicine, mHealth utilizes smartphones and wearable devices for remote consultations and health monitoring.

A notable contribution is the ICT-based KiORH (Kiosk Operated Rural Health-care) application, developed to address the healthcare challenges in resource-limited rural areas. Trained health assistants, use attached sensors and Android devices to collect patient data such as blood pressure and SpO2, which is then uploaded to the cloud for analysis by the doctors sitting in the urban areas. The application supports real-time sensor data transmission via Bluetooth, role-based secure access, and a multilingual interface, including Bengali. This scalable solution significantly improves health-care delivery in rural areas of developing countries like India.

Chapter 4 discusses the design and deployment of a novel data transmission model for video and text, using SMS over cellular network and mobile ad hoc networks (MANETs). This alternative communication method is crucial when internet connectivity is unreliable, especially in rural areas or during disasters. The chapter details new techniques developed to improve video streaming over MANETs, focusing on applications like video surveillance, where fewer pixel changes occur, making the system more efficient.

Additionally, the KiORH application captures patient symptoms and transmits the data to doctors using the internet or SMS, particularly in low-connectivity areas. To optimize data transmission, the LZ77 and LZ78 compression algorithms are employed, reducing health data from over 3000 characters to 160, ensuring efficient delivery of the vitals of the patient via SMS.

Chapter 5 addresses one major challenge of health-care services, that is storage and retrieval of heath data. In this chapter, various Big Data tools and technologies are evaluated, and their effectiveness for storage and retrieval of health-care data are identified. Several tools have been used for evaluation. Hadoop, a distributed processing framework, supports the storage and processing of vast datasets across clusters. Hive offers SQL-like querying for easier data analysis, while HBase provides real-time read/write capabilities on Hadoop. NoSQL databases like MongoDB and Cassandra bring scalability and flexibility to the storage of diverse data types. Despite these advancements, the chapter identifies specific limitations in the adoption of Big Data in health-care, particularly regarding efficient query processing.

Chapter 6 introduces a new scheme called Node-Guided MapReduce (NGMR), designed to address the limitations of existing distributed systems like Hadoop. NGMR aims to reduce query response time by executing the query in the selective Datanodes in distributed environments. Unlike traditional MapReduce, NGMR allows selective participation of nodes in execution of queries based on user-defined criteria.

The application built using NGMR was deployed on both a Raspberry Pi and a high-end cluster, demonstrating its ability to decrease retrieval and execution times

for Big Data files. By fragmenting data horizontally, NGMR ensures efficient data processing, which is especially beneficial in large-scale systems such as health-care databases.

In Chapter 7, the performance of NGMR is compared with other Big Data tools, focusing on the storage and retrieval of healthcare data. The chapter also explains the process of mapping a healthcare data model based on EHR standards onto various Big Data solutions, including Hadoop, Hive, HBase, Cassandra, and MongoDB.

The comparative analysis shows that MongoDB performs best for structured data, while Cassandra excels in handling unstructured data. However, NGMR consistently outperforms other tools due to its use of a flat file system and multilevel indexing. The chapter concludes with recommendations for future research, including exploring additional tools like Apache Pig and Graph Databases for health-care data storage and retrieval.

Chapter 8 expands on the efficiency of NGMR by introducing an automated node selection algorithm. Traditional MapReduce does not selectively choose nodes, leading to increase of query execution times. NGMR's automated node selection is based on several factors, such as, CPU, memory, storage, and network health. The proposed algorithm calculates a health score for each node, ensuring that only the healthiest nodes are chosen for the incoming query execution.

This thesis proposes several novel algorithms and an application that incorporates various innovative features for remote health-care, enabling the storage, retrieval, and transmission of remote health data. The proposed algorithms are evaluated using synthetic test data generated from real-world data. The evaluation demonstrates significant advancements in both health-care and disaster management.

9.3 Limitation of current work

The thesis presents significant contributions towards improving health-care data management and services in rural areas through the KiORH application and related technologies like cross-layered video transmission, but certain limitations remain.

In this section, some of the key limitations encountered during the research, are listed below.

- KiORH has been developed and deployed in some parts of rural areas in India but the integration of distributed storage with the KiORH application is still left. The large volumes of health data are managed and stored separately outside the application environment.

- Cross-layer fragmented video transmission was successfully tested in smaller environments, but could not be deployed in a large-scale network. The evaluation of its performance in broader geographical coverage was not performed. Further research is needed to optimize the performance for large area networks.

- Although the performance of the various tools was analyzed for health-care data storage and retrieval, it was not been tested or scaled with significantly larger datasets, in the range of terabytes or petabytes due to the unavailability of the resources. This kind of research is needed to handle the rapid growth of the volume of health-care data.

- The node selection algorithm discussed in Chapter 8 has been tested only with microservices. A proper testbed could have been set up in order to demonstrate the algorithm in real-life scenario.

- The KiORH application is not equipped with any intelligent services made using machine learning or artificial intelligence. So, enhancing the knowledge base is a bit manual effort. As a result, advanced decision support or personalized health-care recommendations are not possible to achieve using KiORH.

- KiORH cannot perform real-time data analytics and immediate decision-making support. Adding real-time analytics would significantly enhance the system's utility.

- KiORH application manages sensitive patient data. The absence of advanced data security measures like end-to-end encryption, and multi-factor authentication makes it prone to vulnerable. Strengthening data security is crucial to ensure that patient information is protected against unauthorized access.

9.4 Outstanding issues

There are some outstanding issues which need further investigation.

- The thesis considered several Big Data tools which have been popularly used during the time period of this research work. However, as more modern tools, like Spark etc are now being used in the industries, research needs to be carried out to study the usefulness of these tools in health-care domain.
- Only a few pre-defined queries have been tested on the Big Data tools. Execution of other types of jobs, including batch jobs needs to be investigated.
- The thesis focuses only on primary health-care services. While health-care is presently a large domain, application of technologies in other health-care services, including early diagnosis, personalized medicine etc could be taken up for research.

9.5 Future work

The conclusion and limitations are discussed in the previous sections. Based on the discussion, some new ideas and opportunities can be formed for further research to enhance the present research work. A list of possible research scopes and directions are described below.

- Enhancement of Node-Guided MapReduce and its integration with health-care systems to improve data processing efficiencies and patient outcomes.
- Improving data security and privacy in remote health-care services to safeguard patient information against unauthorized access.

- Thorough analysis and justification of tool performance differences between traditional MapReduce and proposed NGMR to identify the core factors that increase the performance efficiency and provide a foundation for further optimization.

- Integration of distributed storage solutions like Hadoop or Cassandra with KiORH significantly enhances data accessibility and reliability. Seamless connections between these storage solutions and the application can ensure data integrity and efficient access across different health-care facilities.

- Deployment of cross-layer fragmented video transmission in large networks to check and improve the reliability and quality of video transmissions critical in remote health.

- Conducting Performance testing and scaling of tools with larger datasets ranging from terabytes to petabytes, is essential for checking the tools' capability to handle increasing amounts of data efficiently, which is crucial in real-world health-care settings.

Appendices

We are moving from a world where healthcare is reactive to a world where healthcare is proactive.

— Dr. Eric Topol, director of the Scripps Translational Science Institute, San Diego, California

A

Assessment of usefulness of KiORH

Contents

A.1 Prelude

After two years of operation of the kiosk-based health-care delivery system, a survey has been conducted among the rural population to asses the acceptance of the KiORH application, in particular the approach taken by it and the sustainability of the kiosks of similar types. The survey has been conducted by the Department of Computer Science and Engineering and School of Mobile Computing and Communication, Jadavpur University to understand the impact of the project. An independent organization was involved to conduct the survey. This survey

purely stands on the basis of the perception of the villagers, i.e. the beneficiaries.

A.2 Survey methodology

For the survey procedure of KiORH, two methods have been used. At first villagers have been stratified in terms of 'Goers' and 'Non-goers' of the kiosk. i.e. the persons who have visited the kiosk at least once and the persons who have never visited. To evaluate the 'Goers' perception, 100 random samples were collected from the registered patient list and face to face interviews were conducted using pre-tested, structured and close-ended questionnaires.

The perception of 'Non-goers' about KiORH was assessed by a random systematic method using an electoral roll of Barrah Village. Thus, as usual, 100 more samples were picked from the list by this method. The same face to face interactive interview process was conducted using a pre-tested, structured and close-ended questionnaire. In both cases, vernacular language for questionnaire was used. The survey work was completed within seven consecutive days without compromising the pre-determined sample frame. After the completion of data collection, computation and analysis was performed by multi-stairs observation using the Statistical Package for Social Sciences (SPSS-15 version)

A.3 Village description

A total of 2,223 families reside in the village Barrah and in its surrounding area. Total population of the village is 10,798 of which 5,578 are male and 5,220 are female. The Total area of the village is nearly about 614.7 hectares [42]. Child population in the village between age 0-6 is 1,490. According to the survey, 99% of the respondents have their own house and they are living in the village since their ancestral time. Most of the houses are in poor condition indicating the economic condition of the people living in the village. The only primary healthcare centre in the village had not been operational for a long time. Thus, the patients need to travel to the nearby towns. The nearest town (Dubrajpur) is 33 km away from

this village and the district headquarter, Suri is almost 50 km far from this area. The villagers are mostly engaged as daily laborers.

A.4 Drinking water and food habit

At the time of investigation, it was noticed that 54% of the respondents depend on tube well as their source of water. In the summer season, groundwater level falls drastically. Then the villagers suffer most to get water from the tube wells. Only 23% households use tap water. Till date, 17% of the respondents are dependent on open well which creates an impact on the villagers' health. For this reason, water-borne diseases are significantly high in this area. 2% of the respondents depend on the submersible motor pump for drinking water. The positive side is that the villagers do not need to go far for drinking water, i.e. at least 70% of the villagers get a source of drinking water within 100-meter distance and among this 70%, 31% respondents have a source of drinking water within their own housing premises. Drinking water sources are represented in Figure A.1.

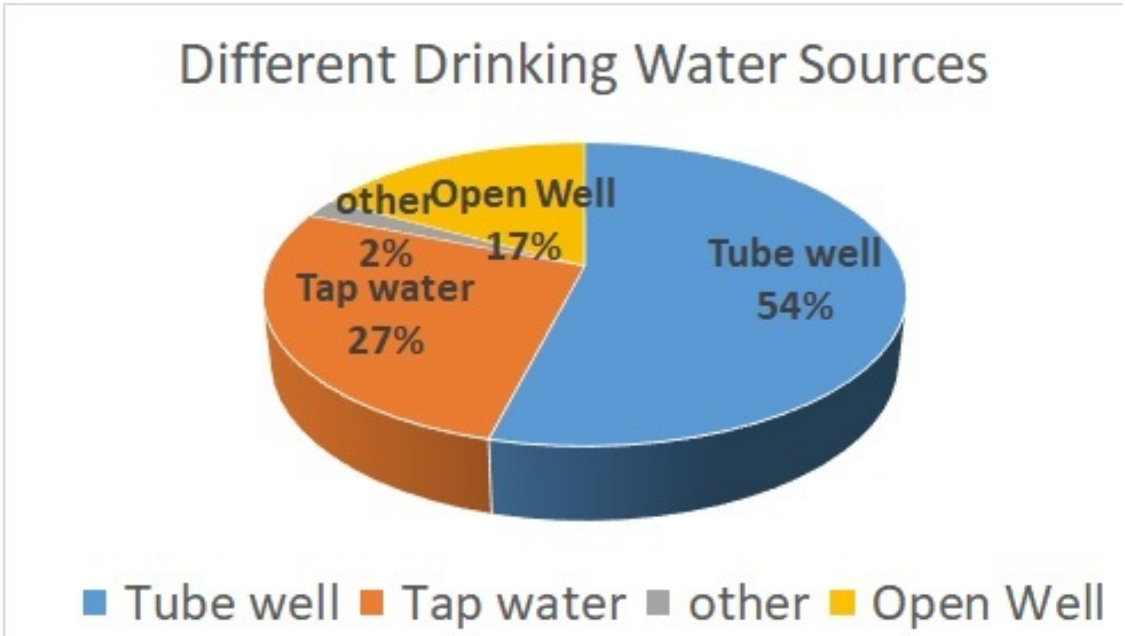

Figure A.1: Drinking water sources in the village

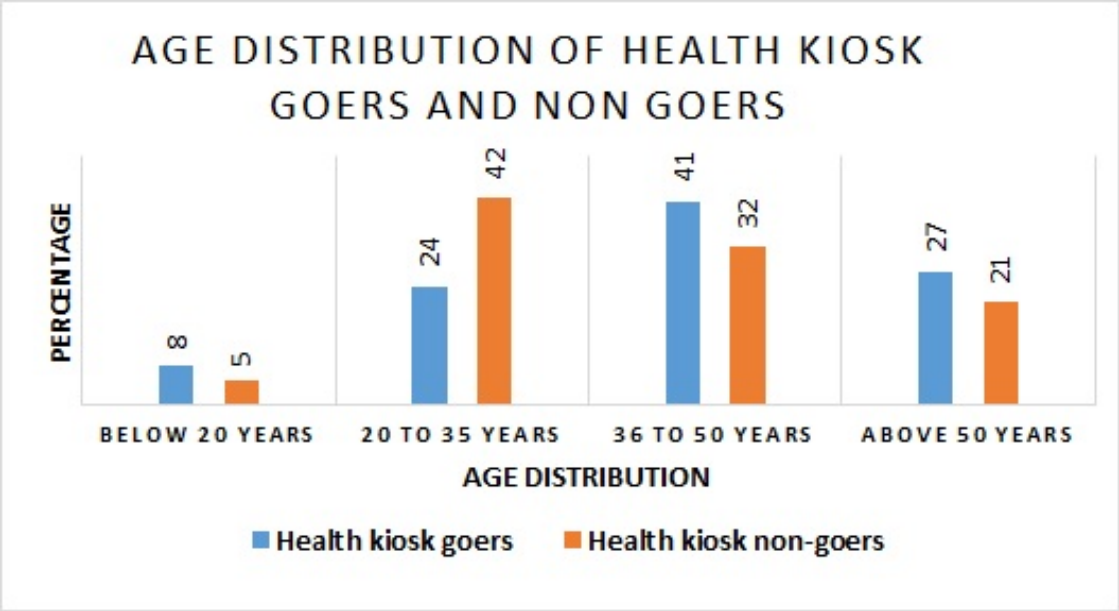

Figure A.2: Age group distribution: goers and non goers to health kiosk

A.5 Survey results

As the survey has been conducted by dividing the respondent into two groups — *health kiosk goers* and *non-goers*, at first the analysis was made by age group. The age distribution of these two groups is shown in Figure A.2. According to the survey, 41% of kiosk goers are in the 35-50 year age group, 24% are in the 24-30 year age group, age of 27% of the population is more than 50 years. Only 8% falls in below 20 years age group. On the other hand, in case of non-goers the numbers are 32%, 42%, 21% and 5% respectively. The results show that older persons (age $>$ 35), particularly persons in the age group 35-50 have utilised the facility more than the younger persons.

A.5.1 Profile of kiosk goers

The first analysis was done based on the responses received from kiosk goers, the respondents who have already experienced the kiosk system at least once. The socio-economic profile of the *health kiosk goers* was analysed first. The analysis was based on gender, religion, caste, education profile, and income of the patient.

- **Gender Distribution:** Gender-wise distribution, shown in Figure A.3, displays the absolute representation in terms of census ratio of the village. The percentage of male patients is slightly high. No gender discrimination was noticed in the analysis.

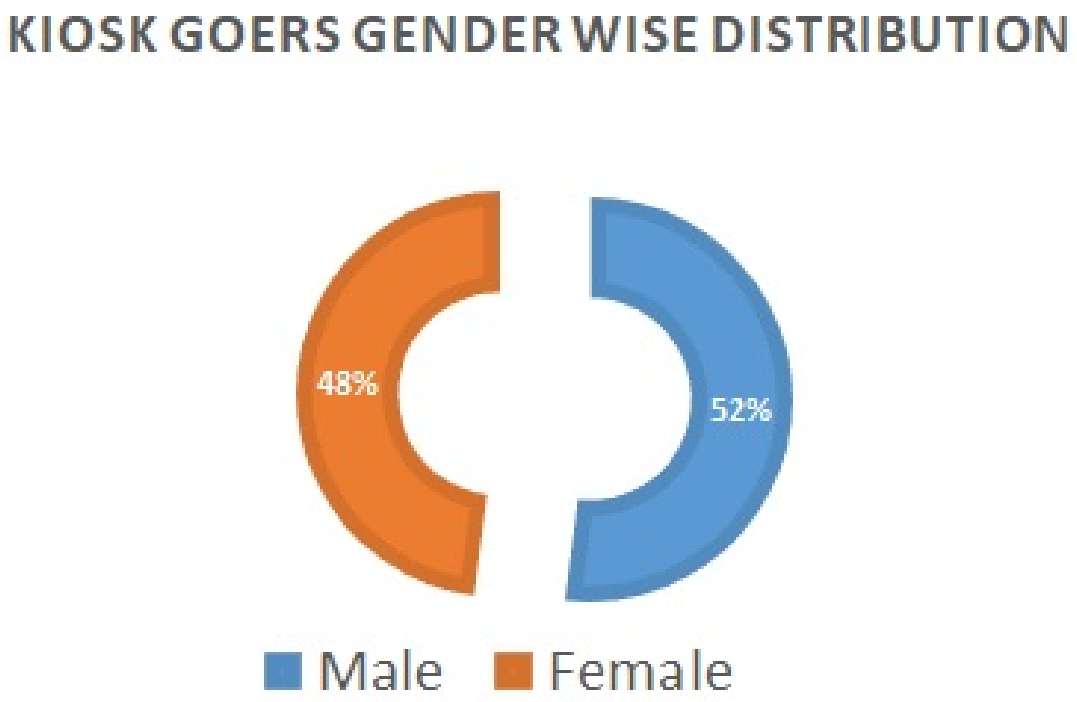

Figure A.3: Gender wise distribution among the patients

- **Religion Distribution:** From religion point of view, the health kiosk goers are distributed in two groups. Hindus represent 58%, followed by Muslims who are 42%. No patient belonging to other religions reside in the village. No religious constraints are noticed from the survey data. Figure A.4 describes the religion distribution.

- **Education Profile:** While analysing the education profile, it is noticed that among the kiosk goers, 65% are graduates. This may be the result of their awareness and hesitation free attitude towards the tab based online health care concept. On the other hand, 29% of the kiosk goers are non-literate. One reason for relying on kiosk-based health-care system in case of these patients may be the geographical location of the kiosk and affordability. They depend

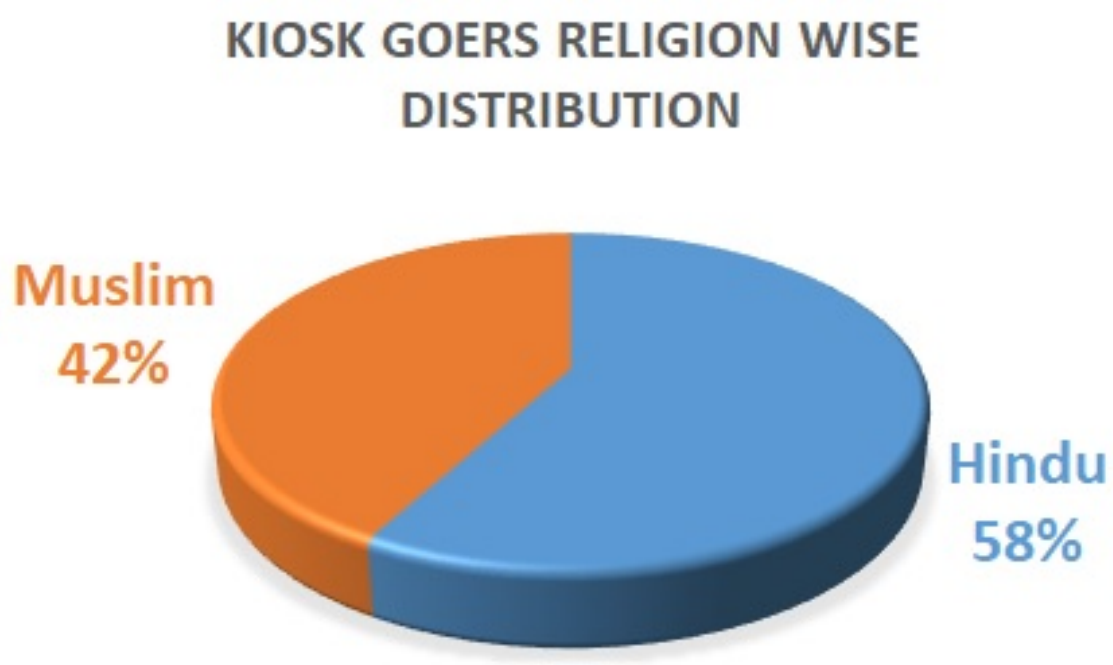

Figure A.4: Religion wise distribution among the patients

on the online cloud-based kiosk system, because they have no provision to go to Dubrajpur or nearby town for treatment. A graphical representation is shown in Figure A.5.

- **Income Group of Patients:** When the patients are divided into income groups, it is observed that 72% of the patients come from low-income group. Riches are usually dependent on either homeopathic medicine or allopathic clinics at the nearby towns. A graphical representation of the distribution of the income groups is shown in Figure A.6.

- **Disease Patterns:** According to the survey findings, it is observed that most of the patients going to the kiosk suffer from chronic diseases. Almost 59% of the sample stated that they go to the kiosks for diseases like diabetes, colitis etc. long-term diseases. Among the kiosk-goers, only 16% and 14% stated that they visited the kiosk for viral and bacterial diseases. Figure A.7 shows the distribution based on disease patterns.

  On the other hand, from another survey record, it is observed that nearly 55% of the patients also depend on other clinics (Allopathic, Homeopathic) for

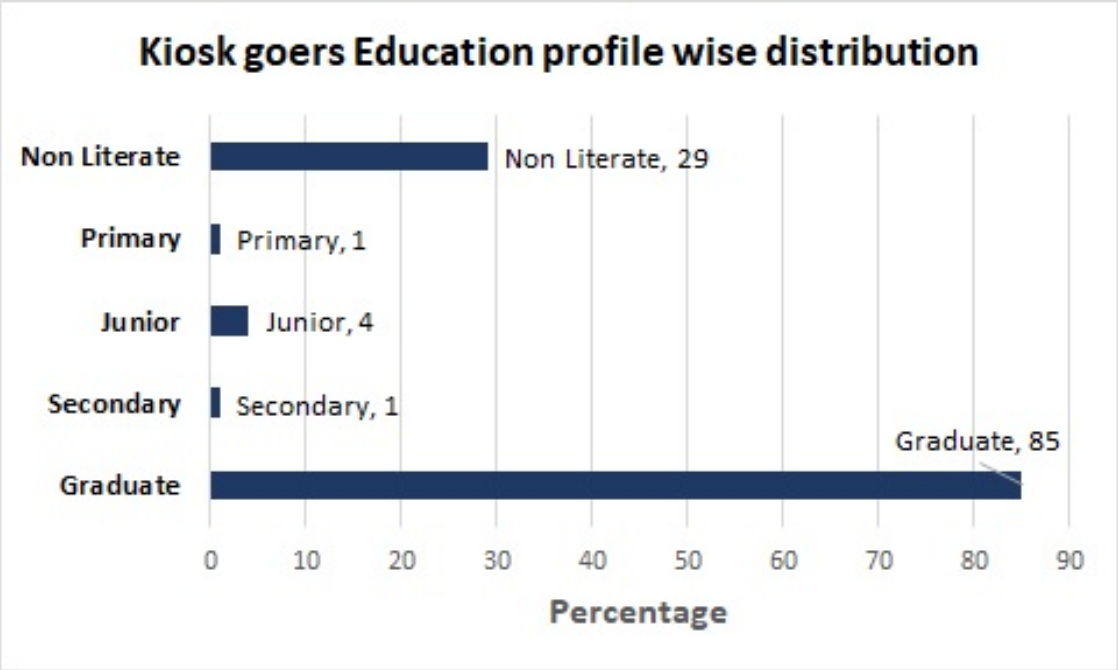

Figure A.5: Education profile of the sample patients

viral and water-borne diseases. They do not have enough confidence to go to the health kiosk of this kind for day to day emergency medical needs, because basic emergency necessities like oxygen or saline water are not available in the kiosk. So most of the kiosk patients visit the kiosk for chronic diseases where there is no emergency needs.

A.5.2 Improvement in patients' health condition

During the survey, patients were also asked about the experiences regarding improvement in their health condition. From the Figure A.8, it is observed that 53% of the patients reported that their health condition improved after their treatment under the health kiosk system. This result is inspiring. However, the remaining 46% of the patients replied negatively when they were asked about the improvement in their health condition after visiting and following the procedure advised by the online doctors in the health kiosk.

The patients mostly agreed that the process followed in the health kiosk system is lengthy, but they did not have complains about the treatment process. 78% realised that the additional time was necessary for this kind of treatment. At

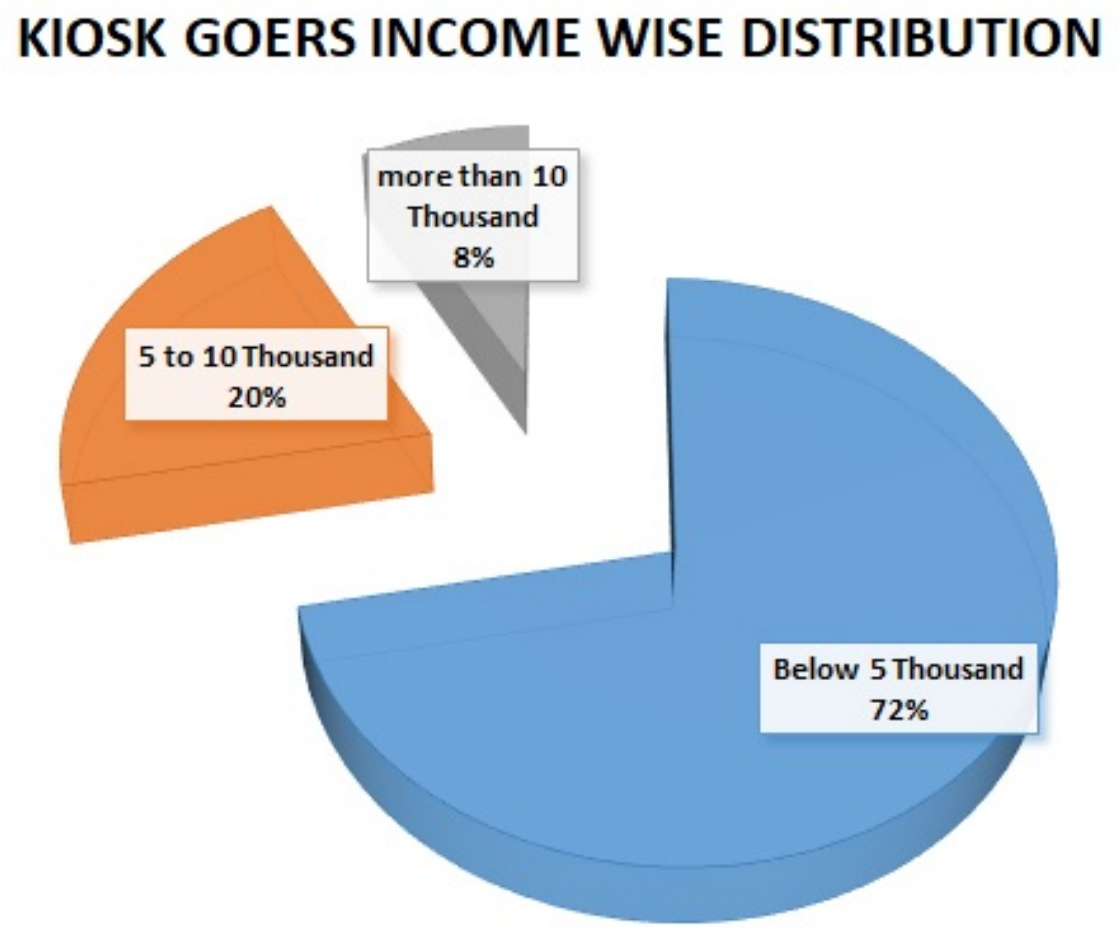

Figure A.6: Income wise distribution among the patients

the time of consultation using KiORH, there is no provision to meet the doctors physically, however, the patient can speak with the doctor through video chat. Almost 76% of the patients had been able to speak with the doctors, and among them, around 85% are satisfied after the video chat. When asked about the reasons for visiting the health kiosk for treatment, the patients have clearly spoken out and mentioned the primary reasons which are shown in Figure A.9.

The health kiosk is geographically close to the villagers, thus, they (24% of patients) think it is beneficiary for them. 34% opine that it is economically affordable and therefore, they depend on it. These two major points came out as positive vibes about health kiosks and can significantly boost-up kiosk-based primary health care in villages where communication and mainstream health care facility is poor. 8% of the patients visit the kiosk because doctors appointed for the health kiosk are really good.

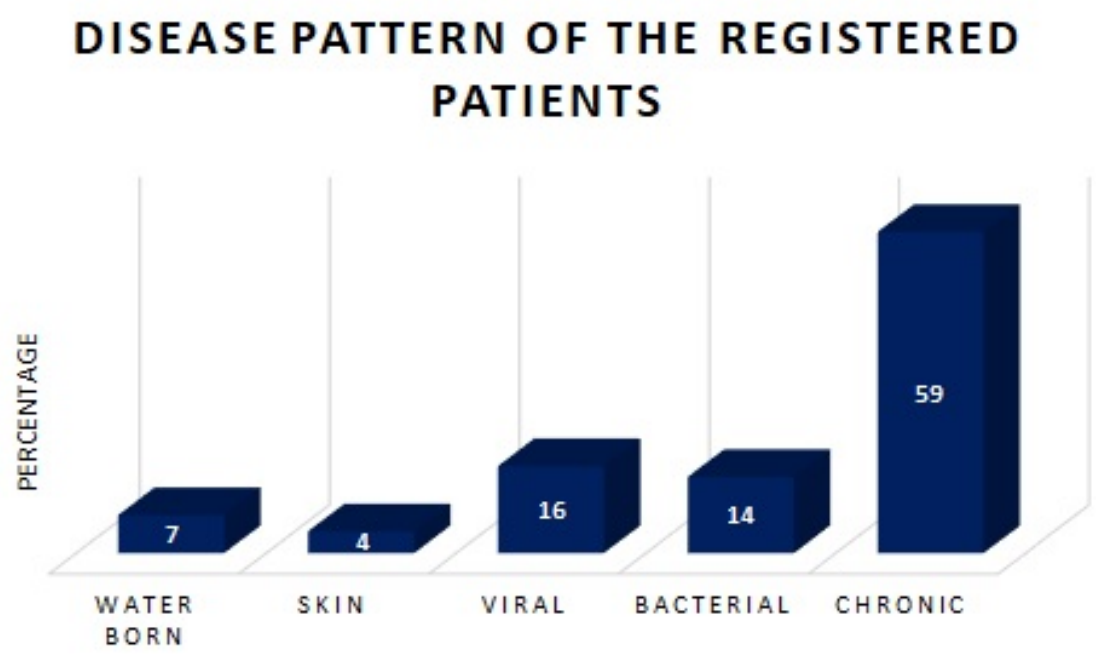

Figure A.7: Disease patterns of the patients

The above-mentioned satisfaction level is confirmed when it is noticed in the Figure A.10. Among the patients, 15% are dissatisfied and said that the service and treatment are not helpful. When they were asked about the reason behind their dissatisfaction, it was found that these patients were mostly from the low-income group and they faced difficulties in buying medicines from the local medicine shop, because either the medicines were not available or expensive. The higher income group does not have much complaints about the medicine cost. Figure A.11 shows the levels of satisfaction among different income groups.

A.5.3 Kiosk non-goers

Next, the survey has been carried out by collecting samples from the villagers who never visited the kiosk. This part of the survey helped to identify the lacuna and drawbacks of the kiosk scheme. The sample may be considered as representative of the village population, but the nature of this set is uncontrolled.

The age group and gender distribution of the population were analysed. Results are shown in Figure A.2 and in Figure A.12 respectively. Regarding gender distribution, it was noticed that the pattern is similar to Census 2011 data.

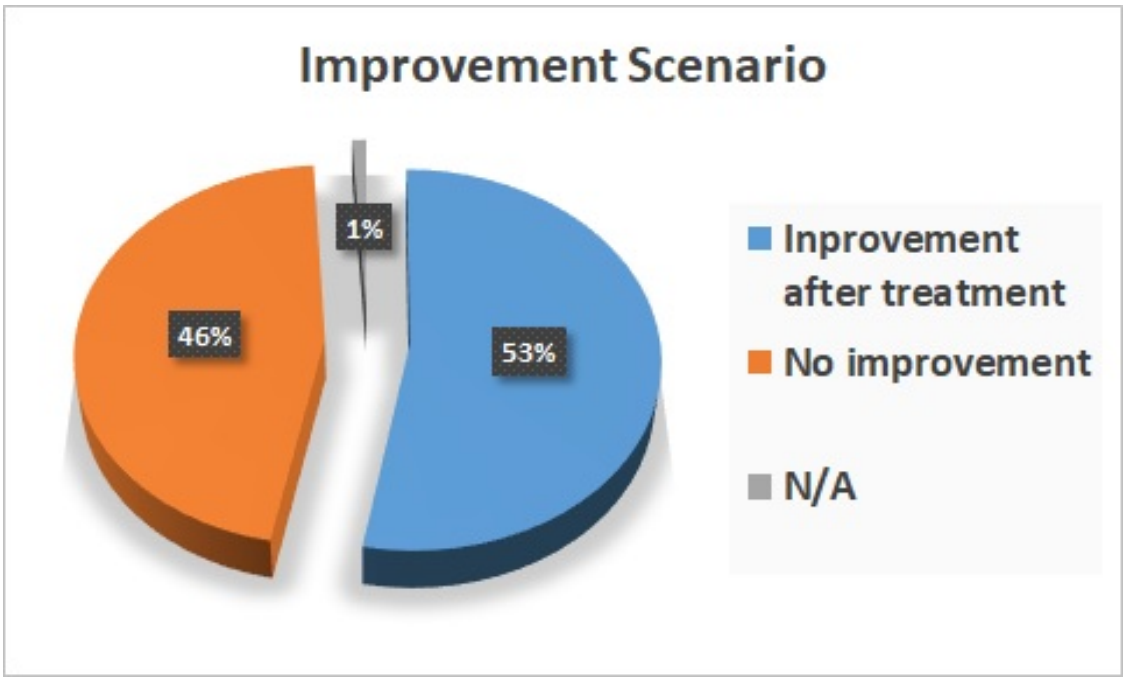

Figure A.8: Health condition Improvement scenario

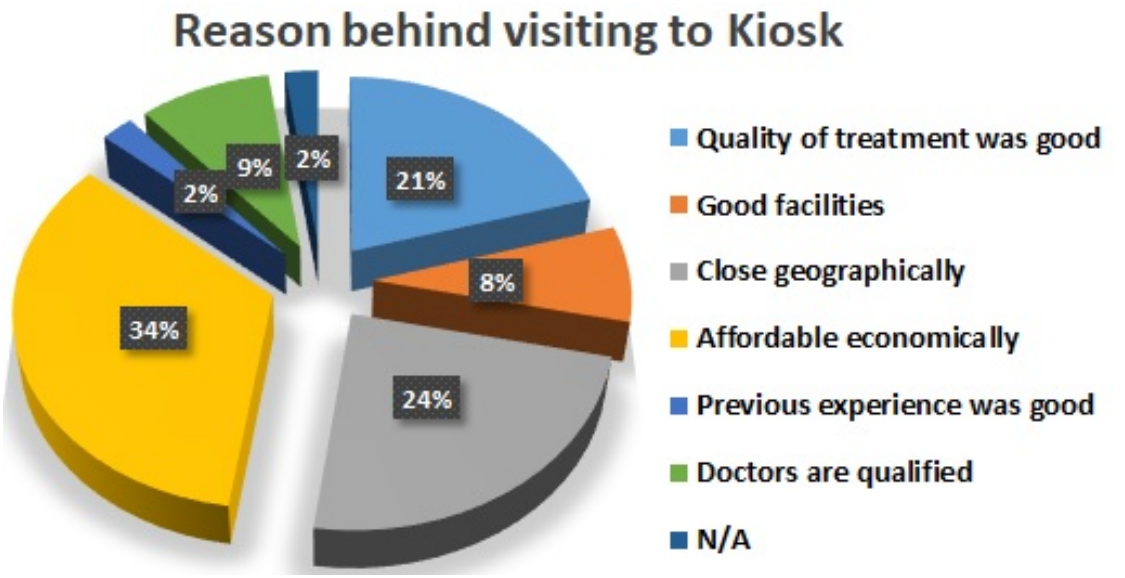

Figure A.9: Reasons behind visiting kiosk

Measuring the general awareness level about the health kiosk among this section of population has been another focus of this survey. It was found that 53% of the villagers recognized the health kiosk. This set of the sample population was further asked why they did not go to the Health kiosk for treatment. Most of the people (almost 44%) said that they did not go there because doctors and nursing staff are

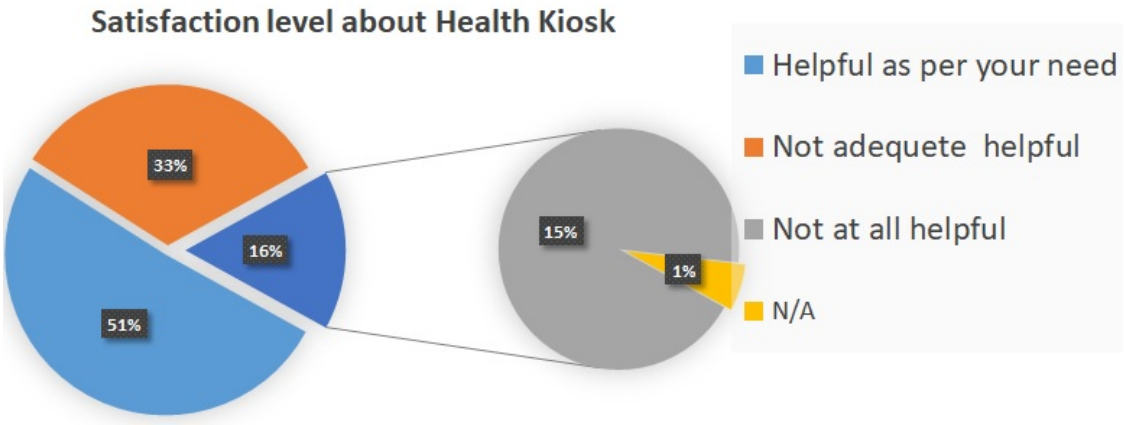

Figure A.10: Satisfaction level among kiosk goers

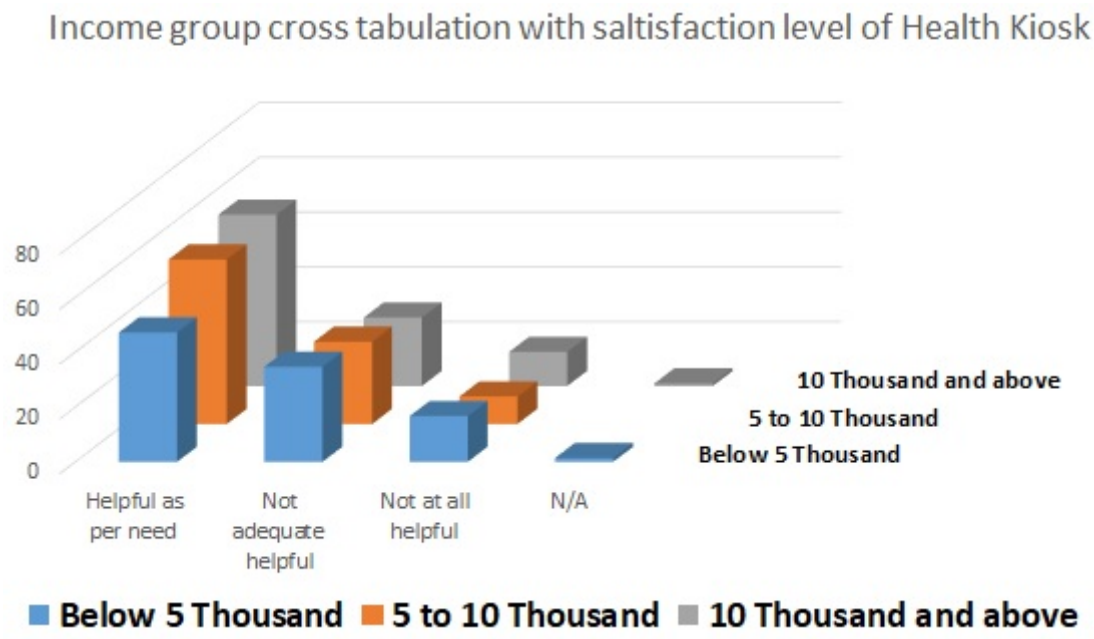

Figure A.11: Income group chart with satisfaction level of the kiosk goers

not available. This kind of treatment, i.e. treatment without the physical presence of a doctor, could not earn the trust of these villagers. Some sizable amount of respondents also complained about the lack of facilities which are found in any usual health center. These two major points helped to understand the views of the non-goer villagers. Other reasons are also shown in Figure A.13.

Apart from this, 52% of the respondents have the knowledge that their neighbors are going to the health kiosk for treatment and 61% stated that their neighbors' experiences were good and gave a positive feedback. Only 3% said that the kiosk is

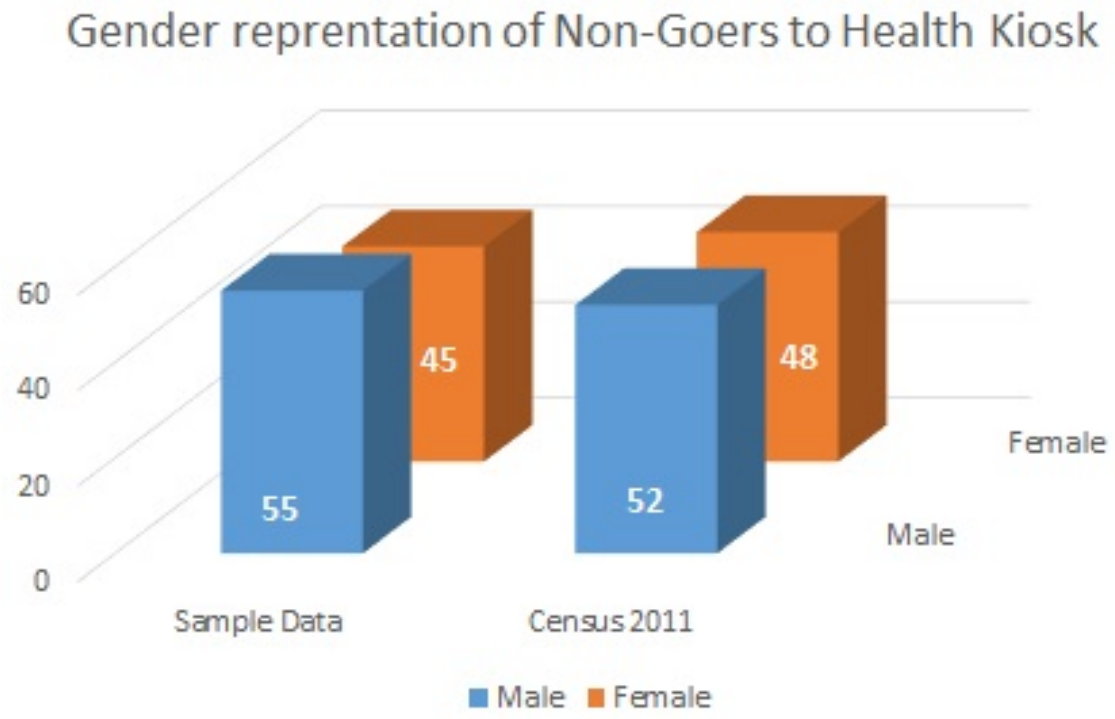

Figure A.12: Gender representation among kiosk non goers

helpful, but not affordable for them. This result is shown in Figure A.14.

A.5.4 Reasons for not re-visiting the kiosk

Further study was made to understand the satisfaction level of the patients who visited the kiosk at least once for treatment. In this regard the respondents' data was analysed and it was noticed that 36% did not visit the kiosk for the second time for a checkup. When they were asked about the reasons, they gave different answers, which are shown in Figure A.15

A.6 Summary and discussion

It has been found that unavailability of the prescribed medicine was one of the strong reasons for not revisiting the kiosk. The patients who stated that the medicines are not available were cross-checked and it was found that they consistently maintained their view in a revised questionnaire. Thus, it was concluded that scarcity or unavailability of medicine was one of the primary issues related to the performance of kiosk-based treatment process.

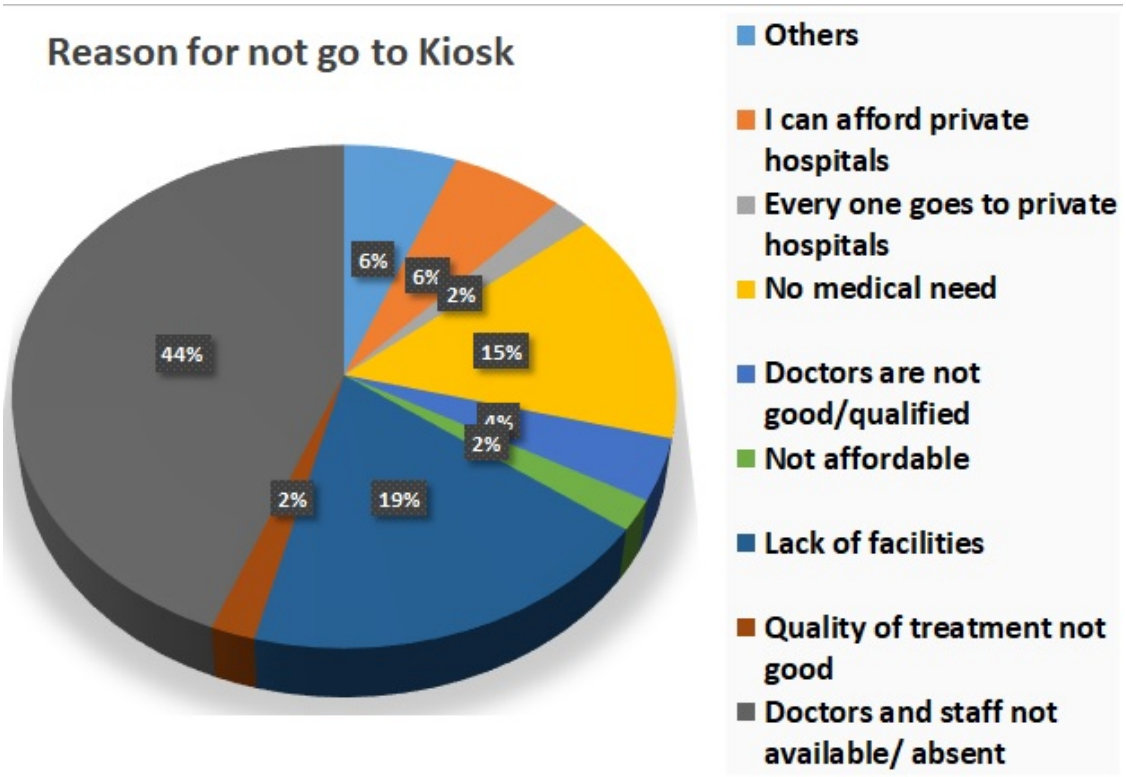

Figure A.13: Reasons for not going to the kiosk

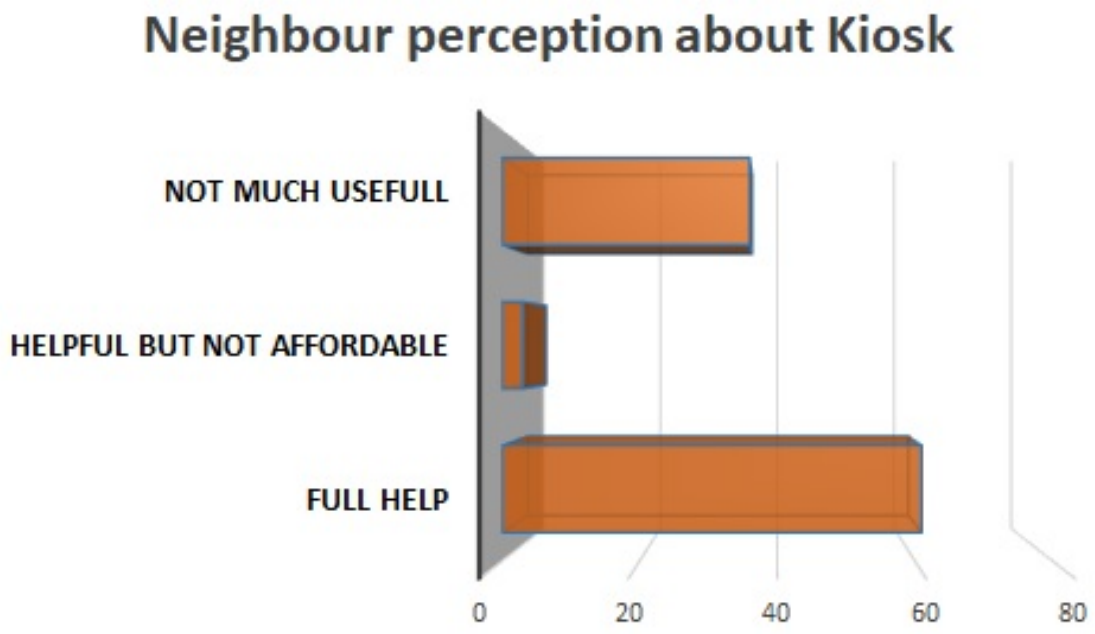

Figure A.14: Neighbors' perception about the health kiosk

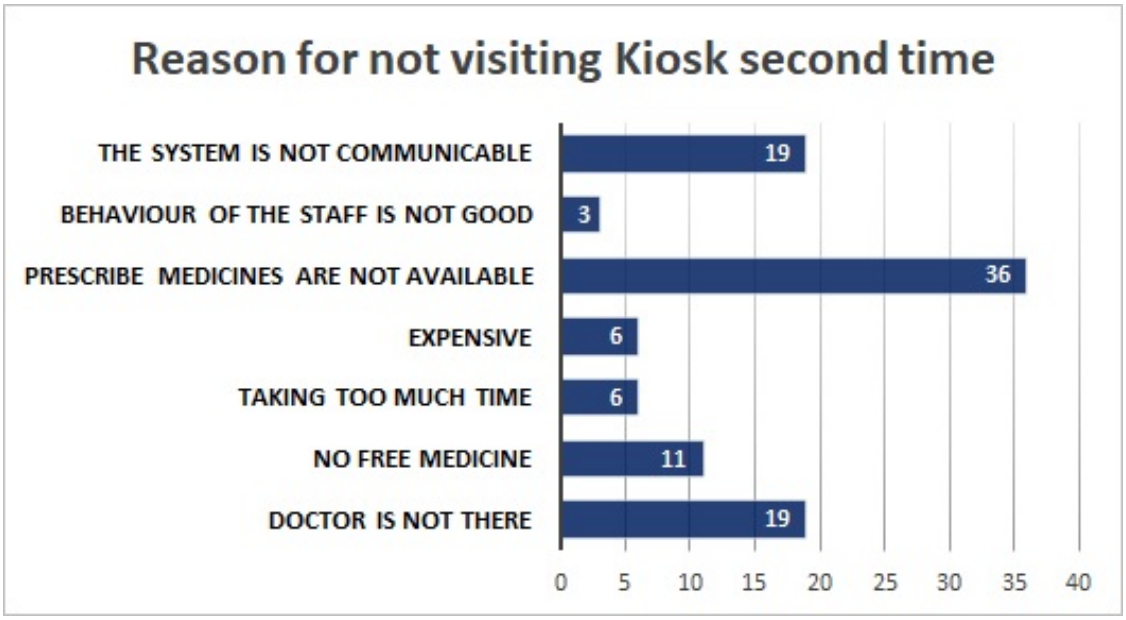

Figure A.15: Reasons behind not visiting the kiosk for the 2nd time

The first kind of intellectual and artistic personality belongs to the hedgehogs, the second to the foxes ...

— Sir Isaiah Berlin [31]

References

[1] URL: http://www.practo.com/.

[2] URL: https://www.credihealth.com/.

[3] URL: https://curofy.com/.

[4] URL: https://www.netmeds.com/.

[5] URL: https://symptomate.com/interview.

[6] URL: http://www.mdcalc.com/.

[7] URL: https://play.google.com/store/apps/details?id=com.medicaljoyworks.prognosisamp;hl=engl=US.

[8] URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/mobility-visualizer.

[9] URL: https://nhm.gov.in/index1.php.

[10] URL: https://www.retailitinsights.com/doc/solohealth-station-0001.

[11] URL: https://www.higi.com/.

[12] URL: https://www.pharma-smart.com/locate-a-pharmacy.

[13] URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/mobility-visualizer.

[14] URL: https://www.pharmaccess.org/update/11312/.

[15] URL: https://www.lalpathlabs.com/..

[16] URL: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance.

[17] URL: https://www.twilio.com/en-us.

[18] 2016. URL: https://www.gadgets360.com/apps/features/roundup-10-indian-healthcare-startups-you-should-know-about-792075.

[19] 2018. URL: https://ehealth.eletsonline.com/2016/05/ehealth-kiosks-to-connect-rural-rajasthan/.

[20] 2019. URL: https://www.healthpointdigital.com.au/about/.

[21] 2023. URL: https://www.electronicsforu.com/india-corner/innovations-innovators/ikure-health-monitoring-kiosks.

[22] Raafat Aburukba et al. "Remote Monitoring Framework for Elderly Care Home Centers in UAE". In: *2020 IEEE International Conference on E-health Networking, Application and Services (HEALTHCOM)*. 2021, pp. 1–6. DOI: 10.1109/HEALTHCOM49281.2021.9398921.

[23] Usman Akhtar et al. “The Impact of Big Data In Healthcare Analytics”. In: *2020 International Conference on Information Networking (ICOIN)*. 2020, pp. 61–63. DOI: 10.1109/ICOIN48656.2020.9016588.

[24] Wafaa S. Albaldawi, Rafah M. Almuttairi, and Mehdi Ebady Manaa. “Big Data Analysis for Healthcare Application using Minhash and Machine Learning in Apache Spark Framework”. In: *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. 2022, pp. 1–7. DOI: 10.1109/HORA55278.2022.9799934.

[25] Folch-Ayora Ana et al. “Mobile applications in oncology: A systematic review of Health Science Databases”. In: *International Journal of Medical Informatics* 133 (2020), p. 104001. DOI: 10.1016/j.ijmedinf.2019.104001.

[26] V G Anisha Gnana Vincy, T Karthija, and J Sunil. “Understanding Hadoop framework through single-node cluster installation”. In: *2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC)*. Nagercoil, India: IEEE, Mar. 2019.

[27] K. Anusha et al. “Comparative Study of MongoDB vs Cassandra in big data analytics”. In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. 2021, pp. 1831–1835. DOI: 10.1109/ICCMC51019.2021.9418441.

[28] Jose Maria A. Araujo et al. “Comparative Performance Analysis of NoSQL Cassandra and MongoDB Databases”. In: *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*. 2021, pp. 1–6. DOI: 10.23919/CISTI52073.2021.9476319.

[29] Hussein Muzahim Aziz, Håkan Grahn, and Lars Lundberg. “Sub-frame crossing for streaming video over wireless networks”. In: *2010 Seventh International Conference on Wireless On-demand Network Systems and Services (WONS)*. 2010, pp. 53–56. DOI: 10.1109/WONS.2010.5437132.

[30] Ayan Banerjee and Sandeep K.S. Gupta. “Analysis of Smart Mobile Applications for Healthcare under Dynamic Context Changes”. In: *IEEE Transactions on Mobile Computing* 14.5 (2015), pp. 904–919. DOI: 10.1109/TMC.2014.2334606.

[31] Isaiah Berlin. *The hedgehog and the fox*. Ed. by Henry Hardy. 2nd ed. Princeton, NJ: Princeton University Press, 2013.

[32] Aditya Bhardwaj et al. “Big Data Emerging Technologies: A CaseStudy with Analyzing Twitter Data using Apache Hive””. In: *Proceedings of 2015 RAECS UIET Panjab University*. IEEE, 2015.

[33] Tian Bo and Jing Chunmei. “A Streaming Transmission Scheme for DVC over Wireless Mobile Ad hoc Networks”. In: *2021 IEEE International Conference on Artificial Intelligence and Industrial Design (AIID)*. 2021, pp. 701–704. DOI: 10.1109/AIID51893.2021.9456493.

[34] Stefan Bosnic, Istvan Papp, and Sebastian Novak. “The development of hybrid mobile applications with Apache Cordova”. In: *2016 24th Telecommunications Forum (TELFOR)*. Belgrade, Serbia: IEEE.

[35] Hicham Boudlal, Mohammed Serrhini, and Ahmed Tahiri. "Cloud Computing for Healthcare Services: Technology, Application and Architecture". In: *2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC)*. 2022, pp. 1–5. DOI: 10.1109/ISIVC54825.2022.9800212.

[36] Harvard Business. "How Big Data Impacts Healthcare by Harvard Business Review Analytics Services". In: (2014).

[37] Toly Chen. "Assessing factors critical to smart technology applications to mobile health care the fgm-fahp approach". In: *Health Policy and Technology* 9.2 (2020), 194–203. DOI: 10.1016/j.hlpt.2020.02.005.

[38] Mike Cottle et al. "Transforming Health Care Through Big Data Strategies for leveraging big data in the health care industry". In: *Institute for Health Technology Transformation, http://ihealthtran. com/big-data-in-healthcare* (2013).

[39] Andrea De Mauro, Marco Greco, and Michele Grimaldi. "What is Big Data? A Consensual Definition and a Review of Key Research Topics". In: Sept. 2014. DOI: 10.13140/2.1.2341.5048.

[40] Jeffrey Dean and Sanjay Ghemawat. "MapReduce". en. In: *Commun. ACM* 51.1 (Jan. 2008), pp. 107–113.

[41] Akon Dey, Alan Fekete, and Uwe Röhm. "REST+T: Scalable Transactions over HTTP". In: *2015 IEEE International Conference on Cloud Engineering*. 2015, pp. 36–41. DOI: 10.1109/IC2E.2015.11.

[42] *District Census Handbook, Birbhum,Village And Town Directory.*

[43] AiLing Duan. "Research and application of distributed parallel search hadoop algorithm". In: *2012 International Conference on Systems and Informatics (ICSAI2012)*. 2012, pp. 2462–2465. DOI: 10.1109/ICSAI.2012.6223552.

[44] Heba Mohammed Fadhil, Haneen Salah Hassan, and Esraa Muneer Abed. "IOT e-health system for cardiac telemonitoring". In: *2022 International Conference on Computer and Applications (ICCA)* (2022). DOI: 10.1109/icca56443.2022.10039546.

[45] Shermal Ruwantha Fernando. URL: https://www.opencv-srf.com/p/introduction.html.

[46] Archana G.K. and V.Deeban Chakravarthy. "HPCA: A node selection and scheduling method for Hadoop MapReduce". In: *2015 International Conference on Computing and Communications Technologies (ICCCT)*. 2015, pp. 368–372. DOI: 10.1109/ICCCT2.2015.7292777.

[47] Ishita Goyal, Amritanshu Singh, and Jaspal Kaur Saini. "Big Data in Healthcare: A Review". In: *2022 1st International Conference on Informatics (ICI)*. 2022, pp. 232–234. DOI: 10.1109/ICI53355.2022.9786918.

[48] Kavya Guntupally et al. "Automated Indexing of Structured Scientific Metadata Using Apache Solr". In: *2020 IEEE International Conference on Big Data (Big Data)*. 2020, pp. 5685–5687. DOI: 10.1109/BigData50022.2020.9378448.

[49] Jonathan Guo and Bin Li. "The application of medical artificial intelligence technology in rural areas of developing countries". en. In: *Health Equity* 2.1 (), pp. 174–181.

[50] Bill Hamilton. “Big data is the future of healthcare”. In: *Cognizant 20-20 insights* (2012).

[51] Jing Han et al. “Survey on NoSQL database”. In: *2011 6th International Conference on Pervasive Computing and Applications*. 2011, pp. 363–366. DOI: 10.1109/ICPCA.2011.6106531.

[52] I. Haratcherev et al. “Link adaptation and cross-layer signaling for wireless video-streaming in a shared medium”. In: *2005 International Conference on Wireless Networks, Communications and Mobile Computing*. Vol. 2. 2005, 1522–1526 vol.2. DOI: 10.1109/WIRLES.2005.1549639.

[53] Alan Hong, Weidong Xiao, and JiangLing Ge. “Big Data Analysis System Based on Cloudera Distribution Hadoop”. In: *2021 7th IEEE Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. 2021, pp. 169–173. DOI: 10.1109/BigDataSecurityHPSCIDS52275.2021.00040.

[54] Yin Huai et al. “Major technical advancements in Apache Hive”. In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (2014). DOI: 10.1145/2588555.2595630.

[55] Nor Khairiah Ibrahim et al. “The performance of video streaming over wireless-N”. In: *2012 IEEE Symposium on Wireless Technology and Applications (ISWTA)*. 2012, pp. 142–145. DOI: 10.1109/ISWTA.2012.6373829.

[56] Olaronke Iroju et al. “Interoperability in healthcare: benefits, challenges and resolutions”. In: *International Journal of Innovation and Applied Studies* 3.1 (2013), pp. 262–270.

[57] M. Jamaluddin and Adhi Dharma Wibawa. “Patient Diagnosis Classification based on Electronic Medical Record using Text Mining and Support Vector Machine”. In: *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*. 2021, pp. 243–248. DOI: 10.1109/iSemantic52711.2021.9573178.

[58] Hung-Chin Jang and Yu-Ti Su. “A Hybrid Design Framework for Video Streaming in IEEE 802.11e Wireless Network”. In: *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*. 2008, pp. 560–567. DOI: 10.1109/AINA.2008.88.

[59] Jetendra Joshi et al. “COMIP: Cluster based overlay and fast handoff mobile IP system for video streaming in VANET’s”. In: *2015 IEEE 3rd International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*. 2015, pp. 1–6. DOI: 10.1109/ICSIMA.2015.7559039.

[60] Severin Kacianka and Hermann Hellwagner. “Adaptive video streaming for UAV Networks”. In: *Proceedings of the 7th ACM International Workshop on Mobile Video* (2015). DOI: 10.1145/2727040.2727043.

[61] Prajwal Bhimrao Kalashetty et al. “AI-Powered Brain Imaging for Early Neurodegenerative Disease Detection”. In: *2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon)*. 2024, pp. 1–6. DOI: 10.1109/MITADTSoCiCon60330.2024.10575866.

[62] P. Kanchanadevi et al. "Cloud-based Protection and Performance Improvement in the E-Health Management Framework". In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020, pp. 268–270. DOI: 10.1109/I-SMAC49090.2020.9243419.

[63] Sindhu Kashyaap. *Kiosk by Kiosk, this healthcare startup is Transforming Healthcare for Rural India*. 2018. URL: https://yourstory.com/2018/06/kiosk-by-kiosk-this-healthcare-startups-is-transforming-healthcare-for-rural-india.

[64] Rahul Katarya and Sajal Jain. "Exploration of Big Data Analytics in Healthcare Analytics". In: *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. 2020, pp. 1–4. DOI: 10.1109/ICCCSP49186.2020.9315192.

[65] Anchal Kathuria and S. N. Panda. "Video capturing and streaming over ad-hoc networks". In: *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)* (2017). DOI: 10.1109/icbdaci.2017.8070867.

[66] Ariane Kerst, Jürgen Zielasek, and Wolfgang Gaebel. "Smartphone applications for depression: A systematic literature review and a survey of health care professionals' attitudes towards their use in clinical practice". In: *European Archives of Psychiatry and Clinical Neuroscience* 270.2 (2019), 139–152. DOI: 10.1007/s00406-018-0974-3.

[67] Gita Khalili Moghaddam and Christopher R. Lowe. "Health and Wellness Measurement Approaches for Mobile Healthcare". In: *SpringerBriefs in Applied Sciences and Technology* (2019). DOI: 10.1007/978-3-030-01557-2.

[68] Yasmeen Khaliq et al. "Calculation of CPU performance, power and cost using Hadoop". In: *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*. 2016, pp. 122–127. DOI: 10.1109/INTECH.2016.7845093.

[69] Mohammad Zunnun Khan et al. "Hadoop based EMH framework: A Big Data approach". In: *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 2021, pp. 1068–1070. DOI: 10.1109/ICACITE51222.2021.9404710.

[70] Zviad Kirtava et al. "E-Health/M-health services for dermatology outpatients screening for skin cancer and follow-up". In: *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)* (2016). DOI: 10.1109/healthcom.2016.7749427.

[71] Yoshihisa Kondo, Hiroyuki Yomo, and Hiroyuki Yokoyama. "Demonstration of Multi-Diversity WLAN Supporting Low-latency and Seamless Video Streaming". In: *2022 IEEE 19th Annual Consumer Communications and Networking Conference (CCNC)*. 2022, pp. 937–938. DOI: 10.1109/CCNC49033.2022.9700496.

[72] L. Lakshmi Prasanna Kumar et al. "Development of Healthcare Architecture based on Cloud Technology and IoT Applications". In: *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. 2023, pp. 1385–1388. DOI: 10.1109/ICCMC56507.2023.10084029.

[73] Muhib Anwar Lambay and S. Pakkir Mohideen. "Big Data Analytics for Healthcare Recommendation Systems". In: *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*. 2020, pp. 1–6. DOI: 10.1109/ICSCAN49426.2020.9262304.

[74] Kai Lei et al. "Research and Application of Query Optimization Based on HBase". In: *2021 7th International Conference on Computer and Communications (ICCC)*. 2021, pp. 1336–1340. DOI: 10.1109/ICCC54389.2021.9674637.

[75] Jian Liu et al. "An Adaptive Cross-Layer Mechanism of Multi-channel Multi-interface Wireless Networks for Real-Time Video Streaming". In: *2010 7th International Conference on Ubiquitous Intelligence and Computing and 7th International Conference on Autonomic and Trusted Computing*. 2010, pp. 165–170. DOI: 10.1109/UIC-ATC.2010.9.

[76] Ligang Liu et al. "Performance Evaluation of BATMAN-Adv Wireless Mesh Network Routing Algorithms". In: *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. 2018, pp. 122–127. DOI: 10.1109/CSCloud/EdgeCom.2018.00030.

[77] Chuntao Lu et al. "The use of mobile health applications to improve patient experience: Cross-sectional study in Chinese Public Hospitals". In: *JMIR mHealth and uHealth* 6.5 (2018). DOI: 10.2196/mhealth.9145.

[78] Evaristus Didik Madyatmadja et al. "Analysis of Big Data in Healthcare Using Decision Tree Algorithm". In: *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)*. Vol. 1. 2021, pp. 313–317. DOI: 10.1109/ICCSAI53272.2021.9609734.

[79] *MoHFW*. URL: https://www.mohfw.gov.in/.

[80] Wai Yin Mok. "A Logical Database Design Methodology for MongoDB NoSQL Databases". In: *2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 2021, pp. 1451–1455. DOI: 10.1109/IEEM50564.2021.9673004.

[81] S Mondal and N Mukherjee. "Mobile-assisted remote healthcare delivery". In: *Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. IEEE, pp. 630–635.

[82] Priyata Mukhopadhyay, Himadri Sekhar Roy, and Nandini Mukherjee. "E-healthcare delivery solution". In: *2019 11th International Conference on Communication Systems and Networks (COMSNETS)*. Bengaluru, India: IEEE.

[83] Kausik Naguri, Poly Sil, and Nandini Mukherjee. "Design of a health-data model and a query-driven implementation in Cassandra". In: *2015 17th International Conference on E-health Networking, Application and Services (HealthCom)*. 2015, pp. 144–148. DOI: 10.1109/HealthCom.2015.7454488.

[84] Jun Ni et al. "Hadoop-Based Distributed Computing Algorithms for Healthcare and Clinic Data Processing". In: *2015 Eighth International Conference on Internet Computing for Science and Engineering (ICICSE)*. 2015, pp. 188–193. DOI: 10.1109/ICICSE.2015.41.

[85] Bun Theang Ong, Komei Sugiura, and Koji Zettsu. "Dynamic pre-training of Deep Recurrent Neural Networks for predicting environmental monitoring data". In: *2014 IEEE International Conference on Big Data (Big Data)*. 2014, pp. 760–765. DOI: 10.1109/BigData.2014.7004302.

[86] Salome Oniani et al. "A review of frameworks on continuous data acquisition for e-health and M-health". In: *Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good* (2019). DOI: 10.1145/3342428.3342702.

[87] Salome Oniani et al. "E-health and M-health applications in Georgia: A review on the free available applications for Android devices". In: *2020 IEEE International Conference on Big Data (Big Data)* (2020). DOI: 10.1109/bigdata50022.2020.9378291.

[88] Vinit Padhye and Anand Tripathi. "Resource Availability Characteristicsand Node Selection in CooperativelyShared Computing Platforms". In: *IEEE Transactions on Parallel and Distributed Systems* 25.4 (2014), pp. 1044–1054. DOI: 10.1109/TPDS.2013.149.

[89] Aditya Patel, Manashvi Birla, and Ushma Nair. "Addressing big data problem using Hadoop and Map Reduce". In: Dec. 2012, pp. 1–5. ISBN: 978-1-4673-1720-7. DOI: 10.1109/NUICONE.2012.6493198.

[90] Maria Patrou et al. "Optimizing Energy Efficiency of Node.js Applications with CPU DVFS Awareness". In: *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*. 2022, pp. 1–8. DOI: 10.1109/IGSC55832.2022.9969367.

[91] Thulani Phakathi et al. "Quality of Service of Video Streaming in Vehicular Adhoc Networks: Performance Analysis". In: *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*. 2016, pp. 886–891. DOI: 10.1109/CSCI.2016.0172.

[92] Eko Sakti Pramukantoro, Dany Primanita Kartikasari, and Reza Andria Siregar. "Performance Evaluation of MongoDB, Cassandra, and HBase for Heterogenous IoT Data Storage". In: *2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI)*. 2019, pp. 203–206. DOI: 10.1109/ICAITI48442.2019.8982159.

[93] Ajay Prasad et al. "A Case Study on the Monitor Mode Passive Capturing of WLAN Packets in an On-the-Move Setup". In: *IEEE Access* 9 (2021), pp. 152408–152420. DOI: 10.1109/ACCESS.2021.3127079.

[94] K Priyanka and N Kulennavar. "a survey on big data analytics in health care". In: *International Journal of Computer Science and Information Technologies* 5 (2014), pp. 5865–5868.

[95] Nadia N. Qadri et al. "Multi-path video streaming with redundant frames over a Manet". In: *Proceedings of the 6th International Conference on Frontiers of Information Technology - FIT '09* (2009). DOI: 10.1145/1838002.1838076.

[96] Wu Qing et al. "A method of pre-sentence text based on Map/Reduce storage and indexing classification". In: *2014 IEEE 5th International Conference on Software Engineering and Service Science*. Beijing, China: IEEE, June 2014.

[97] R. Rathidevi and R. Parameswari. “Performance Analysis of Small Files in HDFS using Clustering Small Files based on Centroid Algorithm”. In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2020, pp. 640–643. DOI: 10.1109/I-SMAC49090.2020.9243418.

[98] Himadri Sekhar Ray et al. “Comparative study of query performance in a remote health framework using Cassandra and Hadoop”. In: *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies*. Rome, Italy: SCITEPRESS - Science, and Technology Publications, 2016.

[99] Kumbala Pradeep Reddy et al. “Machine Learning Revolution in Early Disease Detection for Healthcare: Advancements, Challenges, and Future Prospects”. In: *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*. 2023, pp. 638–643. DOI: 10.1109/ICCCMLA58983.2023.10346963.

[100] Nurul Faizah Rozy et al. “Performance comparison routing protocol AODV, DSDV, and AOMDV with video streaming in Manet”. In: *2019 7th International Conference on Cyber and IT Service Management (CITSM)* (2019). DOI: 10.1109/citsm47753.2019.8965386.

[101] O Rusu et al. “Converting unstructured and semi-structured data into knowledge”. In: *2013 11th RoEduNet International Conference*. Sinaia: IEEE, Jan. 2013.

[102] Himadri Sekhar Ray, Sunanda Bose, and Nandini Mukherjee. “Cloud-based remote healthcare delivery and its impact on society A case study”. In: *Recent Advancements in ICT Infrastructure and Applications* (2022), 249–272. DOI: 10.1007/978-981-19-2374-6_11.

[103] Poly Sil Sen and Nandini Mukherjee. “Standards of EHR and their scope of implementation in a sensor-cloud environment: In Indian context”. In: *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)*. 2014, pp. 241–246. DOI: 10.1109/MedCom.2014.7006011.

[104] Dwi Cahya Setyawan, Tien Fabrianti Kusumasari, and Ekky Novriza Alam. “Data Cleansing Processing using Pentaho Data Integration: Case Study Data Deduplication”. In: *2020 6th International Conference on Science and Technology (ICST)*. Vol. 1. 2020, pp. 01–05. DOI: 10.1109/ICST50505.2020.9732824.

[105] Jeff Shafer, Scott Rixner, and Alan Cox. “The Hadoop distributed filesystem: Balancing portability and performance”. In: Mar. 2010, pp. 122–133. DOI: 10.1109/ISPASS.2010.5452045.

[106] Tarek R. Sheltami. “Performance Evaluation of H.264 Protocol in Ad Hoc Networks”. In: *J. Mob. Multimed.* 4.1 (2008), 59–70. ISSN: 1550-4646.

[107] Li Shu et al. “A Novel Node Selection Method for Real-Time Collaborative Computation in Cloud”. In: *2016 International Conference on Advanced Cloud and Big Data (CBD)*. 2016, pp. 98–103. DOI: 10.1109/CBD.2016.027.

[108] Tshiamo Sigwele et al. “Intelligent and Energy Efficient Mobile Smartphone Gateway for Healthcare Smart Devices Based on 5G”. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–7. DOI: 10.1109/GLOCOM.2018.8648031.

[109] Harmeet Singh et al. "Empirical Investigation of Big Data Analytical Tools: Comparative Analysis". In: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. 2019, pp. 1264–1269. DOI: 10.1109/ICOEI.2019.8862739.

[110] G. Spina et al. "CRNTC+: A smartphone-based sensor processing framework for prototyping personal healthcare applications". In: *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*. 2013, pp. 252–255. DOI: 10.4108/icst.pervasivehealth.2013.252039.

[111] Shangliao Sun. *India: Smartphone users 2040*. 2023. URL: https://www.statista.com/statistics/467163/forecast-of-smartphone-users-in-india/.

[112] K. Takahata, N. Uchida, and Y. Shibata. "Optimal data rate control for video stream transmission over wireless network". In: *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004.* Vol. 1. 2004, 340–345 Vol.1. DOI: 10.1109/AINA.2004.1283934.

[113] Ashish Thusoo et al. "Hive-a petabyte scale data warehouse using hadoop". In: *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE, pp. 996–1005.

[114] Anubha Verma, Harsh Dhand, and Abhijit Shaha. "Healthcare kiosk next generation accessible healthcare solution". In: *HealthCom 2008 - 10th International Conference on e-health Networking, Applications and Services*. Singapore: IEEE.

[115] Guoxi Wang and Jianfeng Tang. "The NoSQL Principles and Basic Application of Cassandra Model". In: *2012 International Conference on Computer Science and Service System*. 2012, pp. 1332–1335. DOI: 10.1109/CSSS.2012.336.

[116] Joe Weinman. "The future of Cloud Computing". In: *2011 IEEE Technology Time Machine Symposium on Technologies Beyond 2020* (2011), pp. 1–2. URL: https://api.semanticscholar.org/CorpusID:41939638.

[117] T. White. *Hadoop: The Definitive Guide*. O'Reilly, 2015. ISBN: 9781491901632. URL: https://books.google.co.in/books?id=CEgGswEACAAJ.

[118] Tom White. *Hadoop: The definitive guide*. O'Reilly and Associates, 2015.

[119] Ahmed E Youssef. "A framework for secure healthcare systems based on big data analytics in mobile cloud computing environments". In: *Int J Ambient Syst Appl* 2.2 (2014), pp. 1–11.

[120] Haiyang Yu et al. "Efficient Continuous Big Data Integrity Checking for Decentralized Storage". In: *IEEE Transactions on Network Science and Engineering* 8.2 (2021), pp. 1658–1673. DOI: 10.1109/TNSE.2021.3068261.

[121] Secil Yuzuk, Murat G Aktas, and Mehmet S Aktas. "On the performance analysis of map-reduce programming model on in-memory NoSQL storage platforms: A case study". In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. ANKARA, Turkey: IEEE, Dec. 2018.

[122] Hancong Zheng et al. "Sustainable Wireless Delivery for HD-Video Streaming via Short Fountain-Code Assisted UDP". In: *2022 IEEE Globecom Workshops (GC Wkshps)*. 2022, pp. 1273–1278. DOI: 10.1109/GCWkshps56602.2022.10008657.

[123] Hui Zhou et al. “Real-Time Video Streaming and Control of Cellular-Connected UAV System: Prototype and Performance Evaluation”. In: *IEEE Wireless Communications Letters* 10.8 (2021), pp. 1657–1661. DOI: 10.1109/LWC.2021.3076415.

[124] Min Zhou, Jiaqi Li, and Jun Ye. “Design of Big Data Compatible Storage System Based on Cloud Computing Environment”. In: *2020 39th Chinese Control Conference (CCC)*. 2020, pp. 3158–3161. DOI: 10.23919/CCC50068.2020.9189477.

[125] Yinghui Zhou et al. “A Healthcare System for Detection and Analysis of Daily Activity Based on Wearable Sensor and Smartphone”. In: *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*. 2015, pp. 1109–1114. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.203.

[126] J Ziv and A Lempel. “A universal algorithm for sequential data compression”. en. In: *IEEE Trans. Inf. Theory* 23.3 (May 1977), pp. 337–343.

[127] J Ziv and A Lempel. “Compression of individual sequences via variable-rate coding”. en. In: *IEEE Trans. Inf. Theory* 24.5 (Sept. 1978), pp. 530–536.

[128] Xiaonan Zou et al. “Logistic Regression Model Optimization and Case Analysis”. In: *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. 2019, pp. 135–139. DOI: 10.1109/ICCSNT47585.2019.8962457.